Introducción al sistema operativo GNU/Linux

Josep Jorba Esteve

P07/M2103/02280



Índice

Introducción

Los sistemas GNU/Linux [Joh98] ya no son una novedad, cuentan con una amplia variedad de usuarios y de ámbitos de trabajo donde son utilizados.

Su origen se remonta al mes de agosto de 1991, cuando un estudiante finlandés llamado Linus Torvalds anunció en una lista de *news* que había creado su propio núcleo de sistema operativo y lo ofrecía a la comunidad de desarrolladores para que lo probara y sugiriera mejoras para hacerlo más utilizable. Éste sería el origen del núcleo (o *kernel*) del operativo que más tarde se llamaría Linux.

Por otra parte, la FSF (Free Software Foundation), mediante su proyecto GNU, producía software (desde 1984) que podía ser utilizado libremente. Debido a lo que Richard Stallman (miembro de la FSF) consideraba software libre, es decir, como aquél del que podíamos conseguir sus fuentes (código), estudiarlas y modificarlas, y redistribuirlo sin que nos obliguen a pagar por ello. En este modelo, el negocio no está en la ocultación del código, sino en el software complementario añadido, en la adecuación del software a los clientes y en los servicios añadidos, como el mantenimiento y la formación de usuarios (el soporte que les demos), ya sea en forma de material, libros y manuales, o en cursos de formación.

La combinación (o suma) del software GNU y del *kernel* Linux, es el que nos ha traído a los actuales sistemas GNU/Linux. Actualmente, los movimientos Open Source, desde diferentes organizaciones (como FSF) y empresas como las que generan las diferentes distribuciones Linux (Red Hat, Mandrake, SuSe...), pasando por grandes empresas como HP, IBM o Sun que proporcionan apoyo, han dado un empujón muy grande a los sistemas GNU/Linux hasta situarlos al nivel de poder competir, y superar, muchas de las soluciones propietarias cerradas existentes.

Los sistemas GNU/Linux no son ya una novedad. El software GNU se inició a mediados de los ochenta, el *kernel* Linux, a principios de los noventa. Y Linux se apoya en tecnología probada de UNIX, con más de 30 años de historia.

En esta unidad introductoria repasaremos algunas ideas generales de los movimientos Open Source y Free software, así como un poco de historia de Linux, y de sus orígenes compartidos con UNIX, de donde ha heredado más de 30 años de investigación en sistemas operativos.

1. Software Libre y Open Source

Bajo la idea de los movimientos (o filosofías) de Software Libre y Open Source [OSIc] [OSIb] (también llamado de código abierto o software abierto), se encuentran varias formas de software, no todas del mismo tipo, pero sí compartiendo muchas ideas comunes.

La denominación de un producto de software como 'de código abierto' conlleva como idea más importante la posibilidad de acceder a su código fuente, y la posibilidad de modificarlo y redistribuirlo de la manera que se considere conveniente, estando sujeto a una determinada licencia de código abierto, que nos da el marco legal.

Frente a un código de tipo propietario, en el cual un fabricante (empresa de software) encierra su código, ocultándolo y restringiéndose los derechos a sí misma, sin dar posibilidad de realizar ninguna adaptación ni cambios que no haya realizado previamente la empresa fabricante, el código abierto ofrece, entre otras consideraciones:

a) Acceso al código fuente, ya sea para estudiarlo (ideal para educación) o modificarlo, sea para corregir errores, adaptarlo o añadir más prestaciones.

b) Gratuidad: normalmente, el software, ya sea en forma binaria o en la forma de código fuente, puede obtenerse libremente o por una módica cantidad en concepto de gastos de empaquetamiento, distribución y valores añadidos.

c) Evitar monopolios de software propietario: no depender de una única opción o único fabricante de nuestro software. Esto es más importante cuando se trata de una gran organización, ya sea una empresa o estado, los cuales no pueden (o no deberían) ponerse en manos de una determinada única solución y pasar a depender exclusivamente de ella.

d) Un modelo de avance, no basado en la ocultación de información, sino en la compartición del conocimiento (semejante al de la comunidad científica), para lograr progresos de forma más rápida, con mejor calidad, ya que las elecciones tomadas están basadas en el consenso de la comunidad, y no en los caprichos de empresas desarrolladoras de software propietario.

Crear programas y distribuirlos junto al código fuente no es nuevo. Ya desde los inicios de la informática y en los inicios de la red Internet se había hecho así. Sin embargo, el concepto de *código abierto* como tal, la definición y la redacción de las condiciones que tenía que cumplir datan de mediados de 1997.

Eric Raymond y Bruce Perens fueron los que divulgaron la idea. Raymond [Ray98] era autor del ensayo titulado "La catedral y el Bazar", que hablaba sobre las técnicas de desarrollo de software utilizadas por la comunidad Linux, encabezada por Linus Torvalds, y la comunidad GNU de la Free Software Foundation (FSF), encabezada por Richard Stallman. Por su parte, Bruce Perens era en aquel momento el jefe del proyecto Debian, que trabajaba en la creación de una distribución de GNU/Linux integrada únicamente con software libre.

Nota

Ver versión española en:
http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html

Nota

Dos de las comunidades más importantes son la FSF, con su proyecto de software GNU, y la comunidad Open Source, cuyo máximo exponente de proyecto es Linux. GNU/Linux es el resultado de la unión de sus trabajos.

Una distinción importante entre estas comunidades son las definiciones de *código abierto* y *software libre*. [Deba] [PS02]

El Software Libre (*Free Software*) [FSF] es un movimiento que parte de las ideas de Richard Stallman, que considera que hay que garantizar que los programas estuviesen al alcance de todo el mundo de forma gratuita, se tuviese acceso libre a éstos y pudieran utilizarse al antojo de cada uno. Una distinción importante, que causó ciertas reticencias a las empresas, es el término *free*. En inglés, este término tiene el doble significado de 'gratuito' y 'libre'. La gente de la FSF buscaba las dos cosas, pero era difícil vender ambas cosas a las empresas; la pregunta típica era: ¿cómo se podía ganar dinero con esto? La respuesta vino de la comunidad Linux (con Linus Torvalds en cabeza), cuando consiguieron tener una cosa que todavía no había logrado la comunidad GNU y la FSF: tener un sistema operativo libre con código fuente disponible. En este momento es cuando a la comunidad se le ocurrió juntar las diversas actividades que había en la filosofía del Software Libre bajo la nueva denominación de *código abierto* (*open source*).

Open Source se registró como una marca de certificación, a la que podían adherirse los productos software que respetasen sus especificaciones. Esto no gustó a todo el mundo y suele haber cierta separación y controversias entre los dos grupos del Open Source y la FSF (con GNU), pero son más las cosas que los unen que las que los separan.

En cierta manera, para los partidarios del software libre (como la FSF), el código abierto (u *open source*) representa un paso en falso, ya que representa una cierta "venta" al mercado de sus ideales, y deja la puerta abierta a que se vaya haciendo propietario el software que era libre. Los partidarios de *open source* ven la oportunidad de promocionar el software que de otra manera estaría en una utilización minoritaria, mientras que con la divulgación y la puesta en común para todo el mundo, incluidas empresas que quieran participar en código abierto, entramos con suficiente fuerza para plantar cara al software propietario.

Sin embargo, la idea que persiguen ambas filosofías es la de aumentar la utilidad del software libre, ofreciendo así una alternativa a las soluciones únicas que las grandes empresas quieren imponer. Las diferencias son más que prácticas.

Una vez establecidas las ideas básicas de la comunidad del código abierto, llegamos al punto en que había que concretar de manera clara qué criterios tenía que cumplir un producto de software para considerarse de código abierto. Había que contar con una definición de código abierto [OSIb], que inicialmente escribió Bruce Perens en junio de 1997 como resultado de comentarios de los desarrolladores de la distribución Debian Linux, y que posteriormente fue reeditada (con modificaciones menores) por la organización OSI (Open Source Initiative). Esta organización está encargada de regular la definición y controlar las licencias de código abierto.

Nota

Ver la definición original de Open Source en:

http://www.opensource.org/docs/definition.php

Y la reedición en:

http://www.opensource.org

Nota

El código abierto está regulado por una definición pública que se utiliza como base de la redacción de sus licencias de software.

Un pequeño resumen (interpretación) de la definición: Un *open source software* [OSIb], o software de código fuente abierto, debe cumplir los requisitos siguientes:

1. Se puede copiar, regalar o vender a terceros el software, sin tener que pagar a nadie por ello. Se permite copiar el programa.

2. El programa debe incluir el código fuente y tiene que permitir la distribución tanto en forma compilada, como en fuente. O, en todo caso, hay que facilitar algún modo de obtener los códigos fuente (por ejemplo, descarga desde Internet). No está permitido ocultar el código o darlo en representaciones intermedias. Garantiza que se pueden hacer modificaciones.

3. La licencia del software tiene que permitir que se puedan realizar modificaciones y trabajos que se deriven, y que entonces se puedan distribuir bajo la misma licencia que la original. Permite reutilizar el código original.

4. Puede requerirse la integridad del código del autor, o sea, las modificaciones se pueden presentar en forma de parches al código original, o se puede pedir que tengan nombres o números distintos a los originales. Esto protege al autor de qué modificaciones puedan considerarse como suyas. Este punto depende de lo que diga la licencia del software.

5. La licencia no debe discriminar a ninguna persona o grupo. No se debe restringir el acceso al software. Un caso aparte son las restricciones por ley, como las de las exportaciones tecnológicas fuera de USA a terceros países. Si existen restricciones de este tipo, hay que mencionarlas.

6. No discriminar campos laborales. El software puede utilizarse en cualquier ambiente de trabajo, aunque no se haya pensado para él. Otra lectura es permitir fines comerciales, nadie puede impedir que el software se utilice con fines comerciales.

7. La licencia es aplicable a todo el mundo que reciba el programa.

8. Si el software forma parte de producto mayor, debe permanecer con la misma licencia. Esto controla que no se separen partes para formar software propietario (de forma no controlada). En el caso de software propietario, hay que informar que hay partes (y cuáles) de software de código abierto.

9. La licencia no debe restringir ningún software incorporado o distribuido conjuntamente, o sea, incorporarlo no debe suponer ninguna barrera para otro producto de software distribuido conjuntamente. Éste es un punto "polémico", ya que parece contradecirse con el anterior, básicamente dice que cualquiera puede coger software de código abierto y añadirlo al suyo sin que afecte a las condiciones de su licencia (por ejemplo propietaria), aunque sí que, según el punto anterior, tendría que informar de que existen partes de código abierto.

10. La licencia tiene que ser tecnológicamente neutra. No deben mencionarse medios de distribución únicos, o excluirse posibilidades. Por ejemplo, no puede limitarse (por licencia) que se haga la distribución en forma de CD, ftp o mediante web.

> Esta definición de *código abierto* no es por sí misma una licencia de software, sino más bien una especificación de qué requisitos debería cumplir una licencia de software de código abierto.

La licencia que traiga el programa tiene que cumplir las especificaciones anteriores para que el programa se considere de código abierto. La organización OSI se encarga de comprobar que las licencias cumplen las especificaciones. En la página web de Open Source Licenses se puede encontrar la lista de las licencias [OSIa], siendo una de las más famosas y utilizadas, la GPL (GNU Public License).

Nota

Open Source Licences:
http://www.opensource.org/licenses/index.html

Bajo GPL, el software puede ser copiado y modificado, pero las modificaciones deben hacerse públicas bajo la misma licencia, y se impide que el código se mezcle con código propietario, para evitar así que el código propietario se haga con partes abiertas. Hay una licencia LGPL que es prácticamente igual, pero permite que software con esta licencia sea integrado en software propietario. Un ejemplo clásico es la biblioteca (*library*) C de Linux (con licencia LGPL); si ésta fuera GPL, sólo podría desarrollarse software libre, con la LGPL se permite usar para desarrollar software propietario.

Muchos proyectos de software libre, o con parte de código abierto y parte propietario, tienen su propia licencia: Apache (basada en BSD), Mozilla (MPL y NPL de Netscape), etc. Básicamente, a la hora de poner el software como *open source* podemos poner nuestra propia licencia que cumpla la definición anterior (de código abierto), o podemos escoger licenciar bajo una licencia ya establecida, o como en el caso de la GPL, nos obliga a que nuestra licencia también sea GPL.

Una vez vistos los conceptos de *código abierto* y sus licencias, nos queda por tratar hasta qué punto es rentable para una empresa trabajar o producir código abierto. Si no fuera atrayente para las empresas, perderíamos a la vez tanto un potencial cliente como uno de los principales productores de software.

El código abierto es también atrayente para las empresas, con un modelo de negocio donde se prima el valor añadido al producto.

En el código abierto existen diferentes rentabilidades atrayentes de cara a las empresas:

a) Para las empresas desarrolladoras de software, se crea un problema, ¿cómo es posible ganar dinero sin vender un producto? Hay mucho dinero gastado en desarrollar un programa y después es necesario obtener beneficios. Bien, la respuesta no es simple, no se puede conseguir con cualquier software, la rentabilidad se encuentra en el tipo de software que puede generar beneficios más allá de la simple venta. Normalmente, hay que hacer un estudio de si la aplicación se tornará rentable al desarrollarla como software abierto (la mayoría sí que lo hará), basándose en las premisas de que tendremos un descenso de gasto en desarrollo (la comunidad nos ayudará), reducción de mantenimiento o corrección de errores (la comunidad puede ofrecer esto muy rápido), y tener en cuenta el aumento de número de usuarios que nos proporcionará el código abierto, así como las necesidades que tendrán de nuestros servicios de apoyo o documentación. Si la balanza es positiva, entonces será viable prescindir de los ingresos generados por las ventas.

b) Aumentar la cuota de usuarios.

c) Obtener mayor flexibilidad de desarrollo, cuantas más personas intervienen, más gente habrá para detectar errores.

d) Los ingresos en su mayor parte vendrán por el lado del apoyo, formación de usuarios y mantenimiento.

e) En empresas que utilizan software, hay que considerar muchos parámetros a la hora de escoger el software para el desarrollo de las tareas, hay que tener en cuenta cosas como: rendimiento, fiabilidad, seguridad, escalabilidad y coste monetario. Y aunque parece que el código abierto ya supone de por sí una

elección por el coste económico, hay que decir que existe software abierto que puede competir con el propietario (o incluso superarlo) en cualquiera de los otros parámetros. Además, hay que vigilar mucho con las opciones o sistemas propietarios de un único fabricante, no podemos depender únicamente de ellos (podemos recordar casos, en otros ámbitos, como los vídeos beta de Sony frente a VHS, o en los PC la arquitectura MicroChannel de IBM). Tenemos que evitar el uso de monopolios con lo que éstos suponen: falta de competencia en los precios, servicios caros, mantenimiento caro, poca (o nula) variedad de opciones, etc.

f) Para los usuarios particulares ofrece gran variedad de software adaptado a tareas comunes, ya que mucho del software ha sido pensado e implementado por personas que querían hacer esas mismas tareas pero no encontraban el software adecuado. Normalmente, en el caso del usuario particular un parámetro muy importante es el coste del software, pero la paradoja es que en el usuario doméstico es donde se hace más uso de software propietario. Normalmente, los usuarios domésticos hacen uso de productos de software con copias ilegales, algunas estadísticas recientes indican índices del 60-70% de copias ilegales domésticas. El usuario siente que sólo por tener el ordenador doméstico PC ya tiene "derecho" a disponer de software para usarlo. En estos casos estamos bajo situaciones "ilegales" que, aunque no han sido perseguidas, pueden serlo en su día, o bien se intentan controlar por sistemas de licencias (o activaciones de productos). Además, esto tiene unos efectos perjudiciales indirectos sobre el software libre, debido a que si los usuarios hacen un uso amplio de software propietario, esto obliga a quien se quiera comunicar con ellos, ya sean bancos, empresas o administraciones públicas, a hacer uso del mismo software propietario, y ellos sí que abonan las licencias a los productos. Una de las "batallas" más importantes para el software libre es la posibilidad de captar a los usuarios domésticos.

Nota

Las copias ilegales domésticas son también denominadas en ocasiones *copias piratas*.

g) Por último, los estados, como caso particular, pueden obtener beneficios importantes del software de código abierto, ya que pueden disponer de software de calidad a precios "ridículos" comparados con el enorme gasto de licencias de software propietario (miles o decenas de miles). Además de que el software de código abierto permite integrar fácilmente a las aplicaciones cuestiones culturales (de cada país) como, por ejemplo, su lengua. Este último caso es bastante problemático, ya que en determinadas regiones, estados pequeños con lengua propia, los fabricantes de software propietario se niegan a adaptar sus aplicaciones, o instan a que se les pague por hacerlo.

2. UNIX. Un poco de historia

Como antecesor de nuestros sistemas GNU/Linux [Sta02], vamos a recordar un poco la historia de UNIX [Sal94] [Lev]. En origen, Linux se pensó como un clon de Minix (una implementación académica de UNIX para PC) y de algunas ideas desarrolladas en los UNIX propietarios; pero, a su vez, se desarrolló en código abierto, y con orientación a los PC domésticos. Veremos, en este apartado dedicado a UNIX y el siguiente dedicado a GNU/Linux, cómo esta evolución nos ha llevado hasta los sistemas GNU/Linux actuales que pueden competir con cualquier UNIX propietario, y que están disponibles para un amplio número de arquitecturas hardware, desde el simple PC hasta los supercomputadores.

Linux puede ser utilizado en un amplio rango de máquinas. En la lista TOP500, pueden encontrarse varios supercomputadores con GNU/Linux (ver lista en sitio web top500.org): por ejemplo, el MareNostrum, en el Barcelona Supercomputing Center, un cluster, diseñado por IBM, con 10240 CPUs PowerPC con sistema operativo GNU/Linux (adaptado para los requisitos de tales máquinas). En las estadísticas de la lista podemos observar que los supercomputadores con GNU/Linux ocupan en general un 75% de la lista.

Nota

Podemos ver la lista TOP500 de los supercomputadores más rápidos en:

http://www.top500.org

UNIX se inició hacia el año 1969 (tenemos ya casi 40 años de historia) en los laboratorios BTL (Bell Telephone Labs) de AT&T. Éstos se acababan de retirar de la participación de un proyecto llamado MULTICS, cuyo objetivo era crear un sistema operativo con el cual un gran ordenador pudiera dar cabida a un millar de usuarios simultáneos. En este proyecto participaban los BTL, General Electric, y el MIT. Pero falló, en parte, por ser demasiado ambicioso para su época.

Mientras se desarrollaba este proyecto, dos ingenieros de los BTL que participaban en MULTICS: Ken Thompson y Dennis Ritchie, encontraron un ordenador que no estaba utilizando nadie, un DEC PDP7, que sólo tenía un ensamblador y un programa cargador. Thompson y Ritchie desarrollaron como pruebas (y a menudo en su tiempo libre) partes de UNIX, un programa ensamblador (del código máquina) y el núcleo rudimentario del sistema operativo.

Ese mismo año, 1969, Thompson tuvo la idea de escribir un sistema de ficheros para el núcleo creado, de manera que se pudiesen almacenar ficheros de forma ordenada en un sistema de directorios jerárquicos. Después de unas cuantas discusiones teóricas (que se alargaron unos dos meses) se implementó el sistema en un par de días. A medida que se avanzaba en el diseño del sistema, en el cual se incorporaron algunos ingenieros más de los BTL, la máquina original se les quedó pequeña, y pensaron en pedir una nueva (en aquellos

días costaban cerca de 100.000 dólares, era una buena inversión). Tuvieron que inventarse una excusa (ya que el sistema UNIX era un desarrollo en tiempo libre) y dijeron que la querían para crear un nuevo procesador de texto (aplicación que daba dinero en aquellos tiempos), y se les aprobó la compra de una PDP11.

UNIX se remonta al año 1969, cuenta con más de 30 años de tecnologías desarrolladas y utilizadas en todo tipo de sistemas.

Cuando les llegó la máquina, sólo les llegó la CPU y la memoria, pero no el disco ni el sistema operativo. Thompson, sin poder esperarse, diseñó un disco RAM en memoria y utilizó la mitad de la memoria como disco, y la otra para el sistema operativo que estaba diseñando. Una vez llegó el disco, se siguió trabajando tanto en UNIX como en el procesador de textos prometido (la excusa). El procesador de textos fue un éxito (se trataba de Troff, un lenguaje de edición, que posteriormente fue utilizado para crear las páginas man de UNIX), y los BTL comenzaron a utilizar el rudimentario UNIX con el nuevo procesador de texto, convirtiéndose así los BTL en el primer usuario de UNIX.

En aquellos momentos comenzaron a presentarse varios principios filosóficos de UNIX [Ray02a]:

- Escribir programas para hacer una cosa y hacerla bien.
- Escribir programas para que trabajaran juntos.
- Escribir programas para que manejaran flujos de texto.

Otra idea muy importante fue que UNIX fue uno de los primeros sistemas pensados para ser independiente de la arquitectura hardware, y que ha permitido portarlo con éxito a un gran número de arquitecturas hardware diferentes.

La necesidad de documentar lo que se estaba haciendo, ya que había usuarios externos, dio lugar en noviembre de 1971 al *UNIX Programmer's Manual*, que firmaron Thompson y Richie. En la segunda edición (junio 1972), denominada V2 (se hacía corresponder la edición de los manuales con el número de versión UNIX), se decía que el número de instalaciones de UNIX ya llegaba a las 10. Y el número siguió creciendo hasta unas 50 en la V5.

Entonces se decidió (finales de 1973) presentar los resultados en un congreso de sistemas operativos. Y como resultado, varios centros informáticos y universidades pidieron copias de UNIX. AT&T no daba apoyo ni mantenimiento de UNIX, lo que hizo que los usuarios necesitaran unirse y compartir sus conocimientos para formar comunidades de usuarios de UNIX. AT&T decidió ceder UNIX a las universidades, pero tampoco les daba apoyo, ni corrección de errores. Los usuarios comenzaron a compartir sus ideas, información de programas, *bugs*, etc. Se creó una asociación denominada USENIX como agrupación de usuarios de UNIX. Su primera reunión (mayo de 1974) tuvo una docena de asistentes.

Nota

Ver: http://www.usenix.org

Una de las universidades que había obtenido una licencia de UNIX fue la Universidad de California en Berkeley, donde había estudiado Ken Thompson. En 1975, Thompson volvió como profesor a Berkeley, y trajo consigo la última versión de UNIX. Dos estudiantes graduados recién incorporados, Chuck Haley y Bill Joy (hoy en día uno de los vicepresidentes de SUN Microsystems) comenzaron a trabajar en una implementación de UNIX.

Una de las primeras cosas que les decepcionó eran los editores; Joy perfeccionó un editor llamado EX, hasta transformarlo en el VI, un editor visual a pantalla completa. Y los dos escribieron un compilador de lenguaje Pascal, que añadieron a UNIX. Hubo cierta demanda de esta implementación de UNIX, y Joy lo comenzó a producir como el BSD, *Berkeley Software Distribution* (o UNIX BSD).

BSD (en 1978) tenía una licencia particular sobre su precio: decía que estaba acorde con el coste de los medios y la distribución que se tenía en ese momento. Así, los nuevos usuarios acababan haciendo algunos cambios o incorporando cosas, vendiendo sus copias "rehechas" y, al cabo de un tiempo, los cambios se incorporaban en la siguiente versión de BSD.

Joy también realizó en su trabajo del editor VI algunas aportaciones más, como el tratamiento de los terminales de texto, de manera que el editor fuera independiente del terminal en que se utilizase; creó el sistema TERMCAP como interfaz genérica de terminales con controladores para cada terminal concreto, de manera que en la realización de los programas ya nos podíamos olvidar de los terminales utilizando la interfaz.

Un siguiente paso fue adaptarlo a diferentes arquitecturas. Hasta el año 1977 sólo se podía ejecutar en máquinas PDP; en ese año se comenzaron a hacer adaptaciones para máquinas del momento como las Interdata e IBM. La versión 7 (V7 en junio 1979) de UNIX fue la primera portable. Esta versión trajo muchos avances, ya que contenía: *awk*, *lint*, *make*, *uucp*; el manual ya tenía 400 páginas (más dos apéndices de 400 cada uno). Se incluía también el compilador de C diseñado en los BTL por Kernighan y Ritchie, que se había creado para reescribir la mayor parte de UNIX, inicialmente en ensamblador y luego pasado a C con las partes de ensamblador que fuesen sólo dependientes de la arquitectura. Se incluyeron también una *shell* mejorada (*shell* de Bourne) y comandos como: *find*, *cpio* y *expr*.

La industria UNIX comenzó también a crecer, empezaron a aparecer versiones (implementaciones) de UNIX por parte de compañías como: Xenix, colaboración entre Microsoft (en los orígenes también trabajó con versiones de UNIX) y SCO para máquinas Intel 8086 (el primer PC de IBM); nuevas versiones BSD de Berkeley...

Pero apareció un nuevo problema, cuando AT&T se dio cuenta de que UNIX era un producto comercial valioso, en la licencia de la V7 se prohibió el estu-

dio en centros académicos, para proteger el secreto comercial. Muchas universidades utilizaban hasta el momento el código fuente de UNIX para docencia de sistemas operativos, y dejaron de usarlo para dar sólo teoría.

Sin embargo, cada uno solucionó el problema a su modo. En Amsterdam, Andrew Tanenbaum (autor de prestigio de libros de teoría de sistema operativos) decidió escribir desde el principio un nuevo sistema operativo compatible con UNIX sin utilizar una sola línea de código de AT&T; llamó a este nuevo operativo Minix. Éste sería el que posteriormente le serviría en 1991 a un estudiante finlandés para crear su propia versión de UNIX, que llamó Linux.

Bill Joy, que continuaba en Berkeley desarrollando BSD (ya estaba en la versión 4.1), decidió marcharse a una nueva empresa llamada SUN Microsystems, en la cual acabó los trabajos del 4.2BSD, que posteriormente modificaría para crear el UNIX de SUN, el SunOS (hacia 1983). Cada empresa comenzó a desarrollar sus versiones: IBM con AIX, DEC con Ultrix, HP con HPUX, Microsoft/SCO con Xenix, etc. UNIX comenzó (desde el año 1980) su andadura comercial, AT&T sacó una ultima versión llamada UNIX SystemV (SV), de la cual derivan, junto con los 4.xBSD, los UNIX actuales, ya sea de la rama BSD o de la SystemV. La SV tuvo varias revisiones, por ejemplo, la SV Release 4 fue una de las más importantes. La consecuencia de estas últimas versiones es que más o menos todos los UNIX existentes se adaptaron uno al otro; en la práctica son versiones del SystemV R4 de AT&T o del BSD de Berkeley, adaptadas por cada fabricante. Algunos fabricantes lo especifican y dicen que su UNIX es de tipo BSD o SV, pero la realidad es que todos tienen un poco de las dos, ya que posteriormente se hicieron varios estándares de UNIX para intentar uniformizarlos; entre ellos encontramos los IEEE POSIX, UNIX97, FHS, etc.

Con el tiempo, UNIX se dividió en varias ramas de sistema, siendo las dos principales, la que derivaba del AT&T UNIX o System V, y la de la universidad de California, el BSD. La mayoría de UNIX actuales derivan de uno u otro, o son una mezcla de los dos.

Pero AT&T en aquellos momentos (SVR4) pasó por un proceso judicial por monopolio telefónico (era la principal, si no la única, compañía telefónica en Estados Unidos), que hizo que se dividiera en múltiples empresas más pequeñas, y los derechos de UNIX originales comenzaron un baile de propietarios importante: en 1990 los tenían a medias el Open Software Foundation (OSF) y UNIX International (UI), después, UNIX Systems Laboratories (USL), que denunció a la Universidad de Berkeley por sus copias del BSD, pero perdió, ya que la licencia original no imponía derechos de propiedad al código de UNIX. Más tarde, los derechos UNIX se vendieron a la empresa Novell, ésta cedió parte a SCO, y hoy en día no está muy claro quién los tiene: por diferentes frentes los reclaman Novell, la OSF y SCO. Un ejemplo reciente de esta problemática puede ser el caso de SCO, que puso una demanda legal a IBM porque ésta, según SCO, había cedido parte del código UNIX a versiones del *kernel* Linux, que

supuestamente incluyen algún código UNIX original. El resultado a día de hoy es que el asunto continúa en los tribunales, con SCO convertida en un "paria" de la industria informática que amenaza a los usuarios Linux, IBM, y otros UNIX propietarios, con la afirmación de que tienen los derechos UNIX originales, y que los demás tienen que pagar por ellos. Habrá que ver cómo evoluciona este caso, y el tema de los derechos UNIX con él.

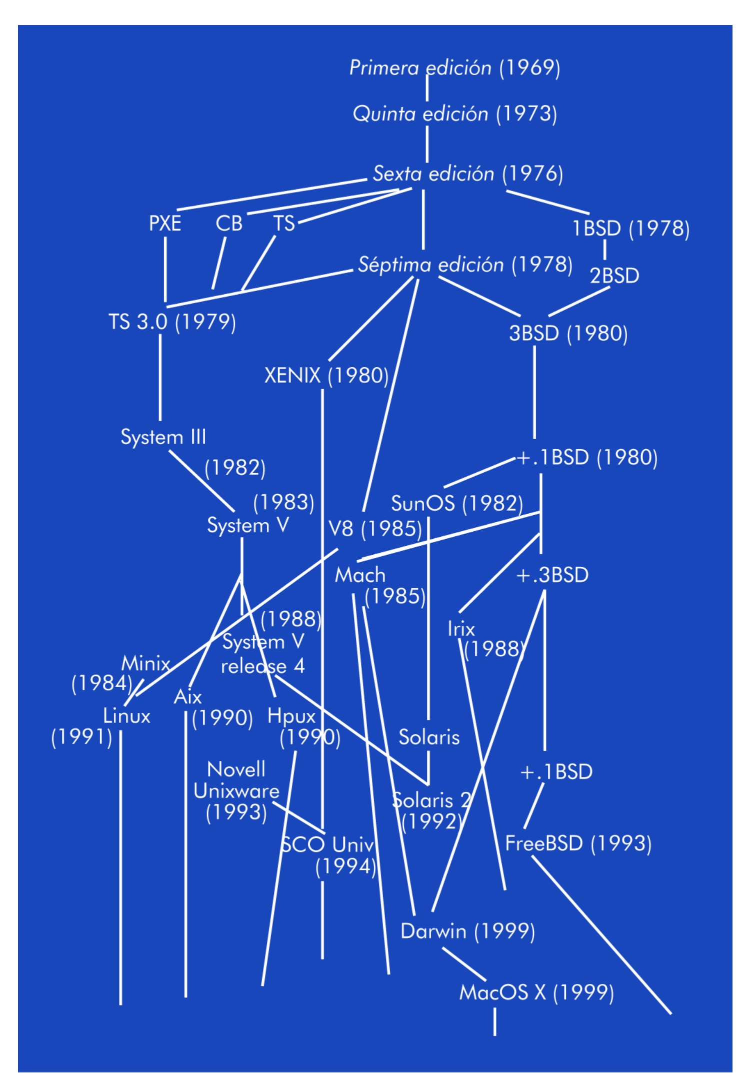


Figura 1. Resumen histórico de varias versiones UNIX

El panorama actual de UNIX ha cambiado mucho desde la aparición de Linux (1991), que a partir de los años 1995-99 comenzó a convertirse en una alternativa seria a los UNIX propietarios, por la gran cantidad de plataformas hardware que soporta y el amplio apoyo de la comunidad internacional y empresas en el avance. Hay diferentes versiones UNIX propietarias que siguen sobreviviendo en el mercado, tanto por su adaptación a entornos industriales o por ser el mejor operativo existente en el mercado, como porque hay necesidades que sólo pueden cubrirse con UNIX y el hardware adecuado. Además, algunos de los UNIX propietarios todavía son mejores que GNU/Linux en cuanto a fiabilidad y rendimiento aunque cada vez acortando distancias, ya que las mismas empresas que tienen sus UNIX propietarios se interesan cada vez más en GNU/Linux, y aportan parte de sus desarrollos para incorporarlos a Linux. Es

de esperar una muerte más o menos lenta de las versiones propietarias de UNIX hacia distribuciones basadas en Linux de los fabricantes adaptadas a sus equipos.

Un panorama general de estas empresas:

Nota

Muchas de las empresas que disponen de UNIX propietarios participan en GNU/Linux y ofrecen algunos de sus desarrollos a la comunidad.

- SUN: dispone de su implementación de UNIX llamada Solaris (evolución del SunOS). Comenzó como un sistema BSD, pero ahora es mayoritariamente SystemV y partes de BSD; es muy utilizado en las máquinas Sun con arquitectura Sparc, y en máquinas multiprocesador (hasta unos 64 procesadores). Promocionan GNU/Linux como entorno de desarrollo para Java, y disponen de una distribución de GNU/Linux denominada Java Desktop System, que ha tenido una amplia aceptación en algunos países. Además, ha comenzado a usar Gnome como escritorio, y ofrece apoyo financiero a varios proyectos como Mozilla, Gnome y OpenOffice. También cabe destacar la iniciativa tomada, con su última versión de su UNIX Solaris, para liberar su código casi totalmente, en la versión Solaris 10. Creando una comunidad para las arquitecturas Intel y Sparc, denominada OpenSolaris, que ha permitido la creación de distribuciones libres de Solaris. Aparte tenemos que señalar iniciativas recientes (2006) para liberar la plataforma Java bajo licencias GPL.

- IBM: tiene su versión de UNIX propietaria denominada AIX, que sobrevive en algunos segmentos de estaciones de trabajo y servidores de la firma. Por otra parte, presta apoyo firme a la comunidad Open Source, promoviendo entornos de desarrollo libres (eclipse.org) y tecnologías Java para Linux, incorpora Linux a sus grandes máquinas y diseña campañas publicitarias (marketing) para promocionar Linux. Además, tiene una repercusión importante en la comunidad por el ambiente judicial de su caso defendiéndose de la firma SCO, que la acusa de violación de propiedad intelectual UNIX, por haber integrado supuestamente componentes en GNU/Linux.

- HP: tiene su UNIX HPUX, pero da amplio soporte a Linux, tanto en forma de código en Open Source, como instalando Linux en sus máquinas. Se dice que es la compañía que ha ganado más dinero con Linux.

- SGI: Silicon Graphics tiene un UNIX llamado IRIX para sus máquinas gráficas, pero últimamente tiende a vender máquinas con Windows, y puede que algunas con Linux. Ha pasado por algunas deficiencias empresariales, por las que estuvo a punto de quebrar. A la comunidad Linux le ofrece soporte de OpenGL (tecnología de gráficos 3D) y de diferentes sistemas de ficheros y control de dispositivos periféricos.

- Apple: se incorporó recientemente (a partir de mediados de los noventa) al mundo UNIX, cuando decidió sustituir su operativo por una variante UNIX. El núcleo llamado Darwin proviene de una versión 4.4BSD; este núcleo Open Source será el que, sumado a unas interfaces gráficas muy potentes, dé a Apple

su sistema operativo MacOS X. Considerado hoy en día como uno de los mejores UNIX y, como mínimo, uno de los más “bellos” en aspecto gráfico. También emplea gran cantidad de software GNU como utilidades de sistema.

- Distribuidores Linux: tanto comerciales como organizaciones, mencionaremos a empresas como Red Hat, SuSe, Mandriva (previamente conocida como Mandrake), y organizaciones no comerciales como Debian, etc. Entre éstas (las distribuciones con mayor despliegue) y las más pequeñas, se llevan el mayor desarrollo de GNU/Linux, y tienen el apoyo de la comunidad Linux y de la FSF con el software GNU, además de recibir contribuciones de las citadas empresas.

- BSD: aunque no sea una empresa como tal, mencionaremos cómo desde Berkeley y otros intermediarios se continúa el desarrollo de las versiones BSD, así como otros proyectos libres clones de BSD como los operativos FreeBSD, netBSD, OpenBSD (el UNIX considerado más seguro), TrustedBSD, etc., que también, más tarde o más temprano, suponen mejoras o incorporaciones de software a Linux. Además, una aportación importante es el *kernel* Darwin proveniente de 4.4BSD, y que desarrolló Apple como núcleo Open Source de su sistema operativo MacOS X.

- Microsoft: aparte de entorpecer el desarrollo de UNIX y GNU/Linux, poniendo trabas con incompatibilidades en diferentes tecnologías, no tiene participación directa en el mundo UNIX/Linux. Aunque en sus orígenes desarrolló Xenix (1980) para PC, a partir de una licencia AT&T de UNIX, que si bien no vendió directamente, sí que lo hizo por medio de intermediarios, como SCO que se hizo con el control en 1987, renombrado como SCO UNIX (1989). Como nota curiosa, posteriormente, compró parte de derechos de la licencia UNIX a SCO (ésta los había obtenido a su vez a través de Novell), no están claros los motivos de Microsoft a la hora de realizar esta adquisición, aunque algunos sugieren que existe alguna relación con el hecho de proporcionar apoyo a SCO en su juicio contra IBM. Además, recientemente (2006), Microsoft llegó a acuerdos con Novell (actual proveedora de la distribución SuSe y la comunidad OpenSuse), en una serie de decisiones bilaterales para promocionar empresarialmente ambas plataformas. Pero parte de la comunidad GNU/Linux se mantiene escéptica, por las posibles implicaciones sobre la propiedad intelectual de Linux y los temas que podrían incluir problemas judiciales por uso de patentes.

Otra anécdota histórica curiosa es que, junto a una empresa llamada UniSys, se dedicaron a realizar marketing de cómo convertir sistemas UNIX a sistemas Windows; y aunque el objetivo podía ser más o menos loable, lo curioso era que el servidor original de la web empresarial estaba en una máquina FreeBSD con Apache. En ocasiones, también paga a algunas empresas “independien-

Nota

Carta abierta de Novell a la comunidad GNU/Linux
http://www.novell.com/linux/microsoft/community_open_letter.html

tes” (algunos opinan que bastante poco) para que lleven a cabo estudios de rendimiento comparativos entre UNIX/Linux y Windows.

Como resumen general, algunos comentarios que suelen aparecer en la bibliografía UNIX apuntan a que: UNIX es técnicamente un sistema sencillo y coherente diseñado con buenas ideas que se supieron llevar a la práctica, pero que no hay que olvidar que algunas de estas ideas se consiguieron gracias al apoyo entusiasta que brindó una gran comunidad de usuarios y desarrolladores que colaboraron entre sí, compartiendo una tecnología y gobernando su evolución.

Y como la historia se suele repetir, en este momento la evolución y el entusiasmo continúan con los sistemas GNU/Linux.

3. Sistemas GNU/Linux

Hace unos veinte años los usuarios de los primeros ordenadores personales no disponían de muchos sistemas operativos donde elegir. El mercado de los ordenadores personales lo dominaba un DOS de Microsoft. Otra posibilidad era los MAC de Apple, pero a unos precios desorbitados (en comparación) con el resto. La otra opción importante, aunque reservada a grandes (y caras) máquinas, era UNIX.

Una primera opción que apareció fue MINIX (1984), creado desde cero por Andrew Tanenbaum, con el objetivo de usarlo para la educación, para enseñar diseño e implementación de sistemas operativos [Tan87] [Tan06].

MINIX fue pensado para ejecutarse sobre una plataforma Intel 8086, muy popular en la época porque era la base de los primeros IBM PC. La principal ventaja de este operativo radicaba en su código fuente, accesible a cualquiera (doce mil líneas de código entre ensamblador y C), ya que estaba incluido en el libro docente de sistemas operativos de Tanenbaum [Tan87]. Pero MINIX era más una herramienta de enseñanza que un sistema eficaz pensado para el rendimiento o para actividades profesionales.

En los noventa, la FSF (Free Software Foundation) y su proyecto GNU, motivó a muchos programadores para promover el software de calidad y de distribución libre. Y aparte de software de utilidades, se trabajaba en un núcleo (*kernel*) de operativo denominado HURD, que llevaría varios años de desarrollo.

Mientras, en octubre de 1991, un estudiante finlandés llamado Linus Torvalds presentaría la versión 0.0.1 de su *kernel* de sistema operativo, que denominó Linux, orientado a máquinas Intel con 386, y lo ofreció bajo licencia GPL a foros de programadores y a la comunidad de Internet para que lo probaran y, si les gustaba, le ayudaran a su desarrollo. El entusiasmo fue tal, que en poco tiempo había un gran número de programadores trabajando en el núcleo o en aplicaciones para él.

Algunas de las características que diferenciaron a Linux de los sistemas de su tiempo y que siguen siendo aplicables, y otras heredadas de UNIX podrían ser:

a) Sistema operativo de código abierto, cualquiera puede disponer de sus fuentes, modificarlas y crear nuevas versiones que poder compartir bajo la licencia GPL (que, de hecho, lo convierte en un software libre).

b) Portabilidad: tal como el UNIX original, Linux está pensado para depender muy poco de una arquitectura concreta de máquina; consecuentemente,

Linux es, en su mayor parte, independiente de la máquina de destino y puede portarse a prácticamente cualquier arquitectura que disponga de un compilador C como el GNU *gcc*. Sólo restan algunas pequeñas partes de código ensamblador y de algunos dispositivos dependientes de la máquina, que tienen que ser rescritas en cada puerto a una nueva arquitectura. Gracias a esto, GNU/Linux es uno de los sistemas operativos que corre en mayor número de arquitecturas: Intel x86 y IA64, AMD x86 y x8664, Sparc de Sun, MIPS de Silicon, PowerPC (Apple), IBM S390, Alpha de Compaq, m68k Motorola, Vax, ARM, HPPArisc...

c) *Kernel* de tipo monolítico: el diseño del *kernel* está unido en una sola pieza, pero es conceptualmente modular en las diferentes tareas. Otra escuela de diseño de operativos propone los *microkernel* (un ejemplo es Mach), donde los servicios se implementan como procesos aparte, comunicados por un (micro) *kernel* más básico. Linux se decidió como monolítico, porque es difícil extraer buen rendimiento de los *microkernels* (es un trabajo bastante duro y complejo). Por otra parte, el problema de los monolíticos es el crecimiento, cuando se vuelven muy grandes se vuelven intratables en el desarrollo, esto se intentó solucionar con los módulos de carga dinámica.

Nota

Proyecto original Mach:
http://www.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html

d) Módulos dinámicamente cargables: permiten poner partes del sistema operativo, como *filesystems*, o *controladores de dispositivos*, como porciones externas que se cargan (o enlazan) con el *kernel* en tiempo de ejecución bajo demanda. Esto permite simplificar el *kernel* y ofrecer estas funcionalidades como elementos que se pueden programar por separado. Con este uso de módulos, se podría considerar a Linux como un *kernel* mixto, ya que es monolítico, pero ofrece una serie de módulos que complementan el *kernel* (aproximación parecida al *microkernel*).

e) Desarrollo del sistema por una comunidad vinculada por Internet: los sistemas operativos nunca habían tenido un desarrollo tan amplio y disperso, no suelen salir de la compañía que los elabora (en el caso propietario) o de un pequeño conjunto de instituciones académicas y laboratorios que colaboran para crear uno. El fenómeno de la comunidad Linux permite que cada uno colabore en la medida en que el tiempo y sus propios conocimientos se lo permitan. El resultado es: de cientos a miles de desarrolladores para Linux. Además, por su naturaleza de sistema de código fuente abierto, Linux es un laboratorio ideal para probar ideas de sistemas operativos al mínimo coste; se puede implementar, probar, tomar medidas y, si funciona, añadir la idea al *kernel*.

Los proyectos se sucedieron y –en el inicio de Linus con el *kernel*– a la gente de la FSF, con el software de utilidad GNU y, sobre todo, con su compilador de C (*GCC*), se les unieron otros proyectos importantes como las XFree (una versión PC de las X Window), los proyectos de escritorio como KDE y Gnome. Y el desarrollo de Internet con proyectos como el servidor web Apache, el navegador Mozilla, o las bases de datos MySQL y PostgreSQL, acabaron por dar al

kernel inicial Linux el recubrimiento de aplicaciones suficiente para construir los sistemas GNU/Linux y competir en igualdad de condiciones con los sistemas propietarios. Y convertir a los sistemas GNU/Linux en el paradigma del software de fuente abierta (Open Source).

Los sistemas GNU/Linux se han convertido en la punta de lanza de la comunidad Open Source, por la cantidad de proyectos que se han podido aglutinar y llevar a buen término.

El nacimiento de nuevas empresas, que crearon distribuciones GNU/Linux (empaquetamientos de *kernel* + aplicaciones) y le dieron apoyo, como Red Hat, Mandrake, SuSe, contribuyó a introducir GNU/Linux en las empresas reacias, y a comenzar el imparable crecimiento que vivimos actualmente.

Comentaremos también la discusión sobre la denominación de los sistemas como GNU/Linux. El término *Linux* para identificar el sistema operativo con que se trabaja es de común uso (para simplificar el nombre), aunque en opinión de algunos desmerece el trabajo de la FSF con el proyecto GNU, el cual ha proporcionado las principales herramientas del sistema. Aun así, el término *Linux*, para referirse al sistema operativo completo es ampliamente usado comercialmente.

Nota

GNU y Linux, por Richard Stallman:

http://www.gnu.org/gnu/linux-andgnu.html.

En general, para seguir una denominación más ajustada a la participación de la comunidad, se utiliza el termino *Linux*, cuando nos estamos refiriendo sólo al núcleo (*kernel*) del sistema operativo. Esto crea cierta confusión, ya que hay gente que habla de “sistemas” o del “sistema operativo Linux” para abreviar. Cuando se trabaja con un sistema operativo GNU/Linux, se está trabajando sobre una serie de software de utilidades, en gran parte fruto del proyecto GNU, sobre el núcleo Linux. Por lo tanto, el sistema es básicamente GNU con un núcleo Linux.

El proyecto GNU de la FSF tenía por objetivo crear un sistema operativo de software libre al estilo UNIX denominado GNU [Sta02].

Linus Torvalds consiguió en 1991 juntar su *kernel* Linux con las utilidades GNU cuando la FSF todavía no disponía de *kernel*. El *kernel* de GNU se denomina HURD, y hoy en día se trabaja bastante en él, y ya existen algunas versiones beta de distribuciones de GNU/HURD (ver más en el apartado dedicado a la administración del *kernel*).

Se calcula que en una distribución GNU/Linux hay un 28% de código GNU y un 3% que corresponde al código del *kernel* Linux; el porcentaje restante corresponde a código de terceros, ya sea de aplicaciones o de utilidades.

Para destacar la contribución de GNU [FSF], podemos ver algunas de sus aportaciones incluidas en los sistemas GNU/Linux:

- El compilador de C y C++ (*GCC*)
- El *shell bash*
- El editor Emacs (GNU Emacs)
- El intérprete *postscript* (*ghostscript*)
- La biblioteca C estándar (*GNU C library*, o también *glibc*)
- El depurador (*GNU gdb*)
- *Makefile* (*GNU make*)
- El ensamblador (*GNU assembler* o *gas*)
- El *linker* (*GNU linker* o *gld*)

Los sistemas GNU/Linux no son los únicos en utilizar software GNU; por ejemplo, los sistemas BSD incorporan también utilidades GNU. Y algunos operativos propietarios como MacOS X (de Apple) también usan software GNU. El proyecto GNU ha producido software de alta calidad, que se ha ido incorporando a la mayor parte de las distribuciones de sistemas basadas en UNIX, tanto libres como propietarias.

Es justo para todo el mundo reconocer el trabajo de cada uno denominando GNU/Linux a los sistemas que trataremos.

4. El perfil del administrador de sistemas

Las grandes empresas y organizaciones dependen cada vez más de sus recursos de computación y de cómo éstos son administrados para adecuarlos a las tareas. El gran incremento de las redes distribuidas, con sus equipos servidores y clientes, ha creado una gran demanda de un nuevo perfil laboral: el llamado *administrador de sistemas*.

El administrador de sistemas tiene una amplia variedad de tareas importantes. Los mejores administradores de sistema suelen ser bastante generalistas, tanto teóricamente como prácticamente. Pueden enfrentarse a tareas como: realizar cableados de instalaciones o reparar cables; instalar sistemas operativos o software de aplicaciones; corregir problemas y errores en los sistemas, tanto hardware como software; formar a los usuarios; ofrecer trucos o técnicas para mejorar la productividad en áreas que pueden ir desde aplicaciones de procesamiento de textos hasta áreas complejas de sistemas CAD o simuladores; evaluar económicamente compras de equipamiento de hardware y software; automatizar un gran número de tareas comunes, e incrementar el rendimiento general del trabajo en su organización.

Puede considerarse al administrador como un perfil de empleado que ayuda a los demás empleados de la organización a aprovechar mejor y más óptimamente los recursos disponibles, de forma que mejore toda la organización.

La relación con los usuarios finales de la organización puede establecerse de diferentes maneras: o bien mediante la formación de usuarios o bien por ayuda directa en el caso de presentarse problemas. El administrador es la persona encargada de que las tecnologías utilizadas por los usuarios funcionen adecuadamente, o sea, que los sistemas cumplan las perspectivas de los usuarios, así como las tareas que éstos quieran realizar.

Hace años, y aún actualmente, en muchas empresas u organizaciones no hay una perspectiva clara del papel del administrador. En los inicios de la informática (años ochenta y noventa) en la empresa, el administrador era visto en un principio como la persona "entendida" en ordenadores (el "gurú") que se encargaba de poner máquinas y que vigilaba o las reparaba en caso de problemas. Normalmente, era una especie de informático polivalente que tenía que solucionar los problemas que fueran apareciendo. Su perfil de currículum no era claro, ya que no necesitaba tener amplios conocimientos, sino sólo tener conocimientos básicos de una decena (como mucho) de aplicaciones (el procesador de texto, la hoja de cálculo, la base de datos, etc.), y algunos conocimientos básicos de hardware eran suficientes para las tareas diarias. Así, cualquier simple "entendido" en el tema podía dedicarse a este trabajo, de manera

que no solían ser informáticos tradicionales, y muchas veces incluso se llegaba a una transmisión oral de los conocimientos entre algún “administrador” más antiguo en la empresa y el nuevo aprendiz.

Con lo anterior, nos encontrábamos de alguna manera en la prehistoria de la administración de sistemas (aunque hay personas que siguen pensando que básicamente se trata del mismo trabajo). Hoy en día, en la época de Internet y de los servicios distribuidos, un administrador de sistemas es un profesional (con dedicación propia y exclusiva) que proporciona servicios en la “arena” del software y hardware de sistemas. El administrador tiene que llevar a cabo varias tareas que tendrán como destino múltiples sistemas informáticos, la mayoría heterogéneos, con objeto de hacerlos operativos para una serie de tareas.

Actualmente, los administradores necesitan tener unos conocimientos generales (teóricos y prácticos) de áreas muy diversas, desde tecnologías de redes, sistemas operativos, aplicaciones de ámbitos diversos, programación básica en una amplia variedad de lenguajes de programación, conocimientos amplios de hardware –tanto del ordenador como de los periféricos usados– tecnologías Internet, diseño de páginas web, bases de datos, etc. Y normalmente también es buscado con el perfil de conocimientos básicos sobre el área de trabajo de la empresa, ya sea química, física, matemáticas, etc. No es de extrañar, entonces, que en una empresa de tamaño medio a grande se haya pasado del “chapuzas” de turno a un pequeño grupo de profesionales con amplios conocimientos, la mayoría con nivel académico universitario, con diferentes tareas asignadas dentro de la organización.

> El administrador debe dominar un rango amplio de tecnologías para poder adaptarse a una multitud de tareas variadas, que pueden surgir dentro de la organización.

Debido a la gran cantidad de conocimientos, no es extraño que aparezcan a su vez diferentes subperfiles de la tarea del administrador. En una gran organización puede ser habitual encontrar a los administradores de sistemas operativos (UNIX, Mac, o Windows), que suelen ser diferentes: administrador de bases de datos, administrador de copias de seguridad, administradores de seguridad informática, administradores encargados de atención a los usuarios, etc.

En una organización más pequeña, varias o todas las tareas pueden estar asignadas a uno o pocos administradores. Los administradores de sistemas UNIX (o de GNU/Linux) serían una parte de estos administradores (cuando no el administrador que tendrá que hacer todas las tareas). Normalmente, su plataforma de trabajo es UNIX (o GNU/Linux en nuestro caso), y requiere de bastantes elementos específicos que hacen este trabajo único. UNIX (y variantes) es un sistema operativo abierto y muy potente, y, como cualquier sistema software, requiere de cierto

nivel de adecuación, configuración y mantenimiento en las tareas para las que vaya a ser usado. Configurar y mantener un sistema operativo es una tarea seria, y en el caso de UNIX puede llegar a ser bastante frustrante.

Algunas áreas importantes por tratar son:

a) Que el sistema sea muy potente también indica que habrá bastantes posibilidades de adaptarlo (configurarlo) a las tareas que queremos hacer. Habrá que evaluar las posibilidades que se nos ofrecen y cuán adecuadas son para nuestro objetivo final.

b) Un sistema abierto y ejemplo claro de ello es nuestro GNU/Linux, que nos ofrecerá actualizaciones permanentes, ya sea en la corrección de errores del sistema, como en la incorporación de nuevas prestaciones. Y, evidentemente, todo esto tiene unos impactos directos importantes en costes de mantenimiento de las tareas de administración.

c) Los sistemas se pueden utilizar para tareas de coste crítico, o en puntos críticos de la organización, donde no se pueden permitir fallos importantes, o que ralenticen o paren la marcha de la organización.

d) Las redes son actualmente un punto muy importante (si no el que más), pero también es un área de problemas potenciales muy crítica, tanto por su propia naturaleza distribuida como por la complejidad del sistema para encontrar, depurar y solucionar los problemas que se puedan presentar.

e) En el caso particular de los sistemas UNIX, y en nuestros GNU/Linux, la abundancia, tanto de versiones como de distribuciones diferentes del sistema, incorpora problemas adicionales a la administración, ya que es necesario conocer las problemáticas y diferencias de cada versión y distribución.

En particular, las tareas de administración del sistema y de la red suelen presentar particularidades diferentes, y a veces se tratan por separado (o por administradores diferentes). Aunque también pueden verse como dos caras del mismo trabajo, con el sistema propiamente dicho (máquina y software) por un lado, y el ambiente donde el sistema (el entorno de red) convive, por el otro.

Normalmente, por *administración de la red* se entiende la gestión del sistema como parte de la red, y hace referencia a los servicios o dispositivos cercanos necesarios para que la máquina funcione en un entorno de red; no cubre dispositivos de red como *switches*, *bridges* o *hubs* u otros dispositivos de red, pero unos conocimientos básicos son imprescindibles para facilitar las tareas de administración.

En este curso trataremos primero aquellos aspectos locales del propio sistema, y en una segunda parte veremos las tareas de administración de red y sus servicios.

Ya hemos comentado el problema de determinar qué es exactamente un administrador de sistemas, ya que en el mercado laboral informático no está demasiado claro. Era común pedir administradores de sistemas según categorías (establecidas en las empresas) de programador o ingenieros de software, las cuales no se adecuan correctamente.

Un programador es básicamente un productor de código; en este caso, un administrador obtendría poca producción, ya que en algunas tareas puede ser necesario, pero en otras no. Normalmente, será deseable que el administrador tenga más o menos conocimientos dependiendo de la categoría laboral:

a) Alguna carrera o diplomatura universitaria, preferible en informática, o en algún campo directamente relacionado con la empresa u organización.

El perfil del administrador suele incluir estudios informáticos o afines a la organización junto con experiencia demostrada en el campo y conocimientos amplios de sistemas heterogéneos y tecnologías de red.

b) Suele pedirse de 1 a 3 años de experiencia como administrador (a no ser que el puesto sea para ayudante de uno ya existente). La experiencia también puede ampliarse de 3 a 5 años.

c) Familiaridad o conocimientos amplios de entornos de red y servicios. Protocolos TCP/IP, servicios de ftp, telnet, ssh, http, nfs, nis, ldap, etc.

d) Conocimientos de lenguajes de *script* para prototipado de herramientas o automatización rápida de tareas (por ejemplo, shell *scripts*, Perl, tcl, Python, etc.) y experiencia en programación de un amplio rango de lenguajes (C, C++, Java, Asm, etc.).

e) Puede pedirse experiencia en desarrollo de aplicaciones grandes en cualquiera de estos lenguajes.

f) Conocimientos amplios de mercado informático, tanto de hardware como de software, en el caso de que haya que evaluar compras de material o montar nuevos sistemas o instalaciones completas.

g) Experiencia en más de una versión de UNIX (o sistemas GNU/Linux), como Solaris, AIX, AT&T SystemV, BSD, etc.

h) Experiencia en sistemas operativos no UNIX, sistemas complementarios que pueden encontrarse en la organización: Windows 9x/NT/2000/XP/Vista, Mac Os, VMS, sistemas IBM, etc.

i) Sólidos conocimientos del diseño e implementación de UNIX, mecanismos de páginas, intercambio, comunicación interproceso, controladores, etc.,

por ejemplo, si las tareas de administración incluyen optimización de sistemas (*tuning*).

j) Conocimientos y experiencia en seguridad informática: construcción de cortafuegos (*firewalls*), sistemas de autentificación, aplicaciones de criptografía, seguridad del sistema de ficheros, herramientas de seguimiento de seguridad, etc.

k) Experiencia en bases de datos, conocimientos de SQL, etc.

l) Instalación y reparación de hardware y/o cableados de red y dispositivos.

5. Tareas del administrador

Según hemos descrito, podríamos separar las tareas de un administrador GNU/Linux (o UNIX en general) [Lev02] en dos partes principales: administración del sistema y administración de red. En los siguientes puntos mostramos de forma resumida en qué consisten en general estas tareas en los sistemas GNU/LINUX (o UNIX); la mayor parte del contenido se va a tratar con cierto detalle en este manual del curso; otra parte, por cuestiones de espacio o complejidad, se explicará superficialmente o no se tratará.

Las tareas de administración engloban una serie de conocimientos y técnicas de los cuales en este curso sólo podemos ver la "punta del *iceberg*"); en todo caso, en la bibliografía adjunta a cada unidad se aportarán referencias para ampliar dichos temas. Como se verá, hay una amplia bibliografía para casi cualquier punto que se trate.

Las tareas de administración del sistema se podrían resumir, por una parte, en la administración local del sistema, y por otra, en la administración de red.

Tareas de administración local del sistema (sin un orden concreto)

- Arranque y apagado del sistema: cualquier sistema basado en UNIX tiene unos sistemas de arranque y apagado valorables, de manera que podemos configurar qué servicios ofrecemos en el arranque de la máquina y cuándo hay que pararlos, o programar el apagado del sistema para su mantenimiento.

- Gestión de usuarios y grupos: dar cabida a los usuarios es una de las principales tareas de cualquier administrador. Habrá que decidir qué usuarios podrán acceder al sistema, de qué forma y bajo qué permisos; y establecer comunidades mediante los grupos. Un caso particular será el de los usuarios de sistema, pseudousuarios dedicados a tareas del sistema.

- Gestión de recursos del sistema: qué ofrecemos, cómo lo ofrecemos y a quién damos acceso.

- Gestión de los sistemas de ficheros: el ordenador puede disponer de diferentes recursos de almacenamiento de datos y dispositivos (disquetes, discos duros, ópticos, etc.) con diferentes sistemas de acceso a los ficheros. Pueden ser permanentes o extraíbles o temporales, con lo cual habrá que modelar y gestionar los procesos de montaje y desmontaje de los sistemas de ficheros que ofrezcan los discos o dispositivos afines.

- Cuotas del sistema: cualquier recurso que vaya a ser compartido tiene que ser administrado, y según la cantidad de usuarios, habrá que establecer un sistema de cuotas para evitar el abuso de los recursos por parte de los usuarios o establecer clases (o grupos) de usuarios diferenciados por mayor o menor uso de recursos. Suelen ser habituales sistemas de cuotas de espacio de disco, o de impresión, o de uso de CPU (tiempo de computación usado).

- Seguridad del sistema: seguridad local, sobre protecciones a los recursos frente a usos indebidos o accesos no permitidos a datos del sistema o de otros usuarios o grupos.

- *Backup* y restauración del sistema: es necesario establecer políticas periódicas (según importancia de los datos), de copias de seguridad de los sistemas. Hay que establecer periodos de copia que permitan salvaguardar nuestros datos, de fallos del sistema (o factores externos) que puedan provocar pérdidas o corrupción de datos.

- Automatización de tareas rutinarias: muchas de las tareas rutinarias de la administración o del uso habitual de la máquina pueden ser fácilmente automatizables, ya debido a su simplicidad (y por lo tanto, a la facilidad de repetirlas), como a su temporalización, que hace que tengan que ser repetidas en periodos concretos. Estas automatizaciones suelen hacerse, bien mediante programación por lenguajes interpretados de tipo *script* (*shells*, Perl, etc.), como por la inclusión en sistemas de temporización (*crontab*, *at*...).

- Gestión de impresión y colas: los sistemas UNIX pueden utilizarse como sistemas de impresión para controlar una o más impresoras conectadas al sistema, así como gestionar las colas de trabajo que los usuarios o aplicaciones puedan enviar a las mismas.

- Gestión de módems y terminales. Estos dispositivos suelen ser habituales en entornos no conectados a red local ni a banda ancha:

  – Los módems permiten una conexión a la red por medio de un intermediario (el ISP o proveedor de acceso), o bien la posibilidad de conectar a nuestro sistema desde el exterior por acceso telefónico desde cualquier punto de la red telefónica.

  – En el caso de los terminales, antes de la introducción de las redes solía ser habitual que la máquina UNIX fuese el elemento central de cómputo, con una serie de terminales "tontos", que únicamente se dedicaban a visualizar la información o a permitir la entrada de información por medio de teclados externos; solía tratarse de terminales de tipo serie o paralelo. Hoy en día, todavía suelen ser habituales en entornos industriales, y en nuestro sistema GNU/Linux de escritorio tenemos una cosa

particular, que son los terminales de texto "virtuales", a los que se accede mediante las teclas Alt+Fxx.

- *Accounting* (o *log*) de sistema: para poder verificar el funcionamiento correcto de nuestro sistema, es necesario llevar políticas de *log* que nos puedan informar de los posibles fallos del sistema o del rendimiento que se obtiene de una aplicación, servicio o recurso hardware. O bien permitir resumir los recursos gastados, los usos realizados o la productividad del sistema en forma de informe.

- *System performance tunning*: técnicas de optimización del sistema para un fin dado. Suele ser habitual que un sistema esté pensado para una tarea concreta y que podamos verificar su funcionamiento adecuado (por ejemplo, mediante *logs*), para examinar sus parámetros y adecuarlos a las prestaciones que se esperan.

- Personalización del sistema: reconfiguración del *kernel*. Los *kernels*, por ejemplo en GNU/Linux, son altamente personalizables, según las características que queramos incluir y el tipo de dispositivos que tengamos o esperemos tener en nuestra máquina, así como los parámetros que afecten al rendimiento del sistema o que consigan las aplicaciones.

Tareas de administración de red

- Interfaz de red y conectividad: el tipo de interfaz de red que utilizamos, ya sea el acceso a una red local, la conexión a una red mayor, o conexiones del tipo banda ancha con tecnologías DSL o RDSI. Además, el tipo de conectividades que vamos a tener, en forma de servicios o peticiones.

- *Routing* de datos: los datos que circularán, de dónde o hacia dónde se dirigirán, dependiendo de los dispositivos de red disponibles y de las funciones de la máquina en red; posiblemente, será necesario redirigir el trafico desde/hacia uno o más sitios.

- Seguridad de red: una red, sobre todo si es abierta (como Internet) a cualquier punto exterior, es una posible fuente de ataques y, por lo tanto, puede comprometer la seguridad de nuestros sistemas o los datos de nuestros usuarios. Hay que protegerse, detectar e impedir posibles ataques con una política de seguridad clara y eficaz.

- Servicios de nombres: en una red hay infinidad de recursos disponibles. Los servicios de nombres nos permiten nombrar objetos (como máquinas y servicios) para poderlos localizar. Con servicios como el DNS, DHCP, LDAP, etc., se nos permitirá localizar servicios o equipos *a posteriori*...

- NIS (*Network Information Service*): las grandes organizaciones han de tener mecanismos para poder organizar de forma efectiva los recursos y el acceso

a ellos. Las formas UNIX estándar, como los *logins* de usuarios con control por *passwords* locales, son efectivos con pocas máquinas y usuarios, pero cuando tenemos grandes organizaciones, con estructuras jerárquicas, usuarios que pueden acceder a múltiples recursos de forma unificada o separada por diferentes permisos,... los métodos UNIX sencillos se muestran claramente insuficientes o imposibles. Entonces se necesitan sistemas más eficaces para controlar toda esta estructura. Servicios como NIS, NIS+, LDAP nos permiten organizar de forma efectiva toda esta complejidad.

- NFS (*Network Fylesystems*): a menudo, en las estructuras de sistemas en red es necesario compartir informaciones (como los propios ficheros) por parte de todos o algunos de los usuarios. O sencillamente, debido a la distribución física de los usuarios, es necesario un acceso a los ficheros desde cualquier punto de la red. Los sistemas de ficheros por red (como NFS) permiten un acceso transparente a los ficheros, independientemente de nuestra situación en la red.

- *UNIX remote commands*: UNIX dispone de comandos transparentes a la red, en el sentido de que, independientemente de la conexión física, es posible ejecutar comandos que muevan información por la red o permitan acceso a algunos servicios de las máquinas. Los comandos suelen tener una "r" delante, con el sentido de 'remoto', por ejemplo: *rcp*, *rlogin*, *rsh*, *rexec*, etc., que permiten las funcionalidades indicadas de forma remota en la red.

- Aplicaciones de red: aplicaciones de conexión a servicios de red, como telnet (acceso interactivo), ftp (transmisión de ficheros), en forma de aplicación cliente que se conecta a un servicio servido desde otra máquina. O bien que nosotros mismos podemos servir con el servidor adecuado: servidor de telnet, servidor ftp, servidor web, etc.

- Impresión remota: acceso a servidores de impresión remotos, ya sea directamente a impresoras remotas o bien a otras máquinas que ofrecen sus impresoras locales. Impresión en red de forma transparente al usuario o aplicación.

- Correo electrónico: uno de los primeros servicios proporcionados por las máquinas UNIX es el servidor de correo, que permite, ya sea el almacenamiento de correo o un punto de retransmisión de correo hacia otros servidores, si no iba dirigido a usuarios propios de su sistema. Para el caso web, también de forma parecida, un sistema UNIX con el servidor web adecuado ofrece una plataforma excelente para web. UNIX tiene la mayor cuota de mercado en cuanto a servidores de correo y web, y es uno de los principales mercados, donde tiene una posición dominante. Los sistemas GNU/Linux ofrecen soluciones de código abierto para correo y web y conforman uno de sus principales usos.

- X Window: un caso particular de interconexión es el sistema gráfico de los sistemas GNU/Linux (y la mayor parte de UNIX), X Window. Este sistema permite una transparencia total de red y funciona bajo modelos cliente servidor; permite que el procesamiento de una aplicación esté desligado de la visualización y de la interacción por medio de dispositivos de entrada, por lo que éstos se sitúan en cualquier parte de la red. Por ejemplo, podemos estar ejecutando una determinada aplicación en una máquina UNIX cuando desde otra visualizamos en pantalla los resultados gráficos, y entramos datos con el teclado y ratón locales de forma remota. Es más, el cliente, llamado cliente X, es tan sólo un componente software que puede ser portado a otros sistemas operativos, permitiendo ejecutar aplicaciones en una máquina UNIX y visualizarlas en cualquier otro sistema. Un caso particular son los llamados terminales X, que son básicamente una especie de terminales "tontos" gráficos que sólo permiten visualizar o interactuar (por teclado y ratón) con una aplicación en ejecución remota.

6. Distribuciones de GNU/Linux

Al hablar de los orígenes de los sistemas GNU/Linux, hemos comprobado que no había un único sistema operativo claramente definido. Por una parte, hay tres elementos software principales que componen un sistema GNU/Linux:

1. El *kernel* Linux: como vimos, el *kernel* es tan sólo la pieza central del sistema. Pero sin las aplicaciones de utilidad, *shells*, compiladores, editores, etc. no podríamos tener un sistema entero.

2. Las aplicaciones GNU: en el desarrollo de Linux, éste se vio complementado con el software de la FSF existente del proyecto GNU, que le aportó editores (como *emacs*), compilador (*gcc*) y utilidades diversas.

3. Software de terceros: normalmente de tipo de código abierto en su mayor parte. Todo sistema GNU/Linux se integra además con software de terceros que permite añadir una serie de aplicaciones de amplio uso, ya sea el propio sistema gráfico de X Windows, servidores como el de Apache para web, navegadores, etc. Asimismo, puede ser habitual incluir algún software propietario, dependiendo del carácter libre que en mayor o menor grado quieran disponer los creadores de la distribución.

Al ser la mayoría del software de tipo de código abierto o libre, ya sea el *kernel*, software GNU o de terceros, normalmente hay una evolución más o menos rápida de versiones, ya sea por medio de corrección de errores o nuevas prestaciones. Esto obliga a que en el caso de querer crear un sistema GNU/Linux, tengamos que escoger qué software queremos instalar en el sistema, y qué versiones concretas de este software.

El mundo GNU/Linux no se limita a una empresa o comunidad particular, con lo que ofrece a cada uno la posibilidad de crear su propio sistema adaptado a sus necesidades.

Normalmente, entre el conjunto de estas versiones siempre se encuentran algunas que son estables, y otras que están en desarrollo, en fases alfa o beta, que pueden tener errores o ser inestables, por lo que habrá que tener cuidado a la hora de crear un sistema GNU/Linux, con la elección de las versiones. Otro problema añadido es la selección de alternativas, el mundo de GNU/Linux es lo suficientemente rico para que haya más de una alternativa para un mismo producto de software. Hay que elegir entre las alternativas posibles, incorporar algunas o todas, si queremos ofrecer al usuario libertad para escoger su software.

Ejemplo

Un caso práctico lo forman los gestores de escritorio de X Window, en que, por ejemplo, nos ofrecen (principalmente) dos entornos de escritorio diferentes como Gnome y KDE; los dos tienen características parecidas y aplicaciones semejantes o complementarias.

En el caso de un distribuidor de sistemas GNU/Linux, ya sea comercial o bien una organización sin beneficio propio, dicho distribuidor tiene como responsabilidad generar un sistema que funcione, seleccionando las mejores versiones y productos software que puedan conseguirse.

En este caso, una distribución GNU/Linux [Dis] es una colección de software que forma un sistema operativo basado en el *kernel* Linux.

Un dato importante a tener en cuenta, y que causa más de una confusión, es que, como cada uno de los paquetes de software de la distribución tendrá su propia versión (independiente de la distribución en que esté ubicado), el número de distribución asignado no mantiene una relación con las versiones de los paquetes software.

El número de distribución sólo sirve para comparar las distribuciones que genera un mismo distribuidor, y no permite comparar entre otras distribuciones. Si queremos hacer comparaciones entre distribuciones, tendremos que examinar los paquetes software principales y sus versiones para poder determinar qué distribución aporta más novedades.

Ejemplo

Pongamos un ejemplo de algunas versiones actuales (las versiones que aparecen se refieren a finales del año 2003):

a) *Kernel* Linux: actualmente podemos encontrar distribuciones que ofrecen uno o más *kernels*, como los de la serie antigua 2.4.x o generalmente los últimos 2.6.x en revisiones (el número x) de diversa actualidad.

b) La opción en el sistema gráfico X Window, en versión de código abierto, que podemos encontrar prácticamente en todos los sistemas GNU/Linux, ya sean algunas versiones residuales de Xfree86 como las que manejan versiones 4.x.y o bien el nuevo proyecto Xorg (siendo un fork del anterior en 2003), que goza de más popularidad en diversas versiones 6.x o 7.x.

c) Gestor de ventanas o escritorio: podemos disponer de Gnome o KDE, o ambos; Gnome con versiones 2.x o KDE 3.x.y.

Podríamos obtener, por ejemplo, una distribución que incluyese *kernel* 2.4, con XFree 4.4 y Gnome 2.14; o bien otra, por ejemplo, *kernel* 2.6, Xorg 6.8,

KDE 3.1. ¿Cuál es mejor?, es difícil compararlas, ya que suponen una mezcla de elementos, y dependiendo de cómo se haga la mezcla, el producto saldrá mejor o peor, y más o menos adaptado a las necesidades del usuario. Normalmente, el distribuidor mantiene un compromiso entre la estabilidad del sistema y la novedad de las versiones incluidas. Así como proporcionar software de aplicación atrayente para los usuarios de la distribución, ya sea éste generalista, o especializado en algún campo concreto.

En general, podría hacerse un mejor análisis de distribuciones a partir de los siguientes apartados, que habría que comprobar en cada una:

a) Versión del núcleo Linux: la versión viene indicada por unos números *X.Y.Z*, donde normalmente *X* es la versión principal, que representa los cambios importantes del núcleo; *Y* es la versión secundaria, y normalmente implica mejoras en las prestaciones del núcleo: *Y* es par en los núcleos estables e impar en los desarrollos o pruebas. Y *Z* es la versión de construcción, que indica el número de la revisión de *X.Y*, en cuanto a parches o correcciones hechas. Los distribuidores no suelen incluir la ultima versión del núcleo, sino la que ellos hayan probado con más frecuencia y puedan verificar que es estable para el software, y componentes, que ellos incluyen. Este esquema de numeración clásico (que se siguió durante las ramas 2.4.x, hasta los inicios de la 2.6), tuvo algunas modificaciones, para adaptarse al hecho de que el kernel (rama 2.6.x) se vuelve más estable, y cada vez las revisiones son menores (significan un salto de versión de los primeros números), pero el desarrollo es continuo y frenético. En los últimos esquemas, se llegan a introducir cuartos números, para especificar de Z cambios menores, o diferentes posibilidades de la revisión (con diferentes parches añadidos). La versión así definida con cuatro números es la que se considera estable (*stable)*. También son usados otros esquemas para las diversas versiones de test (normalmente no recomendables para entornos de producción), como sufijos -rc (*release candidate*), los -mm, que son kernels experimentales con pruebas de diferentes técnicas, o los -git, que son una especie de "foto" diaria del desarrollo del kernel. Estos esquemas de numeración están en constante cambio para adaptarse a la forma de trabajar de la comunidad del kernel, y a sus necesidades para acelerar el desarrollo del kernel.

b) Formato de empaquetado: es el mecanismo empleado para instalar y administrar el software de la distribución. Se suele conocer por el formato de los paquetes de software soportados. En este caso suelen estar los formatos RPM, DEB, tar.gz, mdk, aunque cada distribución suele tener posibilidad de utilizar varios formatos, suele tener uno por defecto. El software acostumbra a venir con sus archivos en un paquete que incluye información sobre su instalación y posibles dependencias con otros paquetes de software. El empaquetado es importante si se usa software de terceros que no venga con la distribución, ya que el software puede encontrarse sólo en algunos sistemas de paquetes, o incluso en uno sólo.

c) Estructura del sistema de archivos: la estructura del sistema de archivos principal (/) nos indica dónde podemos encontrar nuestros archivos (o los propios del sistema) en el *filesystem*. En GNU/Linux y UNIX hay algunos estándares de colocación de los archivos (como veremos en la unidad de herramientas), como por ejemplo el FHS (*filesystem hierarchy standard*) [Lin03b]. Así, si tenemos una idea del estándar, sabremos dónde encontrar la mayor parte de los archivos; luego depende de que la distribución lo siga más o menos y de que nos avisen de los cambios que hayan hecho.

d) *Scripts* de arranque del sistema: los sistemas UNIX y GNU/Linux incorporan unos guiones de arranque (o *shell scripts*) que indican cómo debe arrancar la máquina y cuál será el proceso (o fases) que se van a seguir, así como lo que deberá hacerse en cada paso. Para este arranque hay dos modelos, los de SysV o BSD (es una diferencia de las dos ramas de UNIX principales); y cada distribución podría escoger uno u otro. Aunque los dos sistemas tienen la misma funcionalidad, son diferentes en los detalles, y esto será importante en los temas de administración (lo veremos en la administración local). En nuestro caso, los sistemas analizados, tanto Fedora como Debian, utilizan el sistema de SysV (será el que veremos en la unidad local), pero hay otras distribuciones como Slackware que utilizan el otro sistema BSD. Y existen determinadas propuestas para nuevas opciones en este aspecto de arranque.

e) Versiones de la biblioteca del sistema: todos los programas (o aplicaciones) que tenemos en el sistema dependen para su ejecución de un número (mayor o menor) de bibliotecas de sistema. Estas bibliotecas, normalmente de dos tipos, ya sean estáticas unidas al programa (archivos *libxxx.a*) o las dinámicas que se cargan en tiempo de ejecución (archivos *libxxx.so*), proporcionan gran cantidad de código de utilidad o de sistema que utilizarán las aplicaciones. La ejecución de una aplicación puede depender de la existencia de unas bibliotecas adecuadas y del número de versión concreto de estas bibliotecas (no es lo recomendable, pero puede suceder). Un caso bastante habitual es la biblioteca GNU C library, la biblioteca estándar de C, también conocida como *glibc*. Puede suceder que una aplicación nos pida que dispongamos de una versión concreta de la *glibc* para poder ejecutarse o compilarse. Es un caso bastante problemático, y por ello, uno de los parámetros que valoran la distribución es conocer qué versión de la *glibc* lleva, y los posibles paquetes adicionales de versiones de compatibilidad con versiones antiguas. El problema aparece al intentar ejecutar o compilar un producto de software muy antiguo en una distribución moderna, o bien un producto de software muy nuevo en una distribución antigua.

El mayor cambio llegó al pasar a una *glibc* 2.0, en que había que recompilar todos los programas para poder ejecutarlos correctamente, y en las diferentes revisiones nuevas de numeración 2.x ha habido algunos cambios menores que podían afectar a alguna aplicación. En muchos casos, los paquetes de software comprueban si se tiene la versión correcta de la *glibc*, o en el mismo nombre mencionan la versión que hay que utilizar (ejemplo: *paquete-xxx-glibc2.rpm*).

f) Escritorio X Window: el sistema X Window es el estándar gráfico para GNU/Linux como visualización de escritorio. Fue desarrollado en el MIT en 1984 y prácticamente todos los UNIX tienen una versión del mismo. Las distribuciones GNU/Linux disponen de diferentes versiones como la Xfree86 o la Xorg. Normalmente, el X Window es una capa gráfica intermedia que confía a otra capa denominada gestor de ventanas la visualización de sus elementos. Además, podemos combinar el gestor de ventanas con utilidades y programas de aplicación variados para formar lo que se denomina un entorno de escritorio.

Linux tiene principalmente dos entornos de escritorio: Gnome y KDE. Cada uno tiene la particularidad de basarse en una biblioteca de componentes propios (los diferentes elementos del entorno como ventanas, botones, listas, etc.): *gtk+* (en Gnome) y *Qt* (en KDE), que son las principales bibliotecas gráficas que se usan para programar aplicaciones en estos entornos. Pero además de estos entornos, hay muchos otros, gestores de ventanas o escritorios: XCFE, Motif, Enlightement, BlackIce, FVWM, etc., de modo que la posibilidad de elección es amplia. Además, cada uno de ellos permite cambiar la apariencia (*look & feel*) de ventanas y componentes al gusto del usuario, o incluso crearse el suyo propio.

g) Software de usuario: software añadido por el distribuidor, en su mayoría de tipo Open Source, para las tareas habituales (o no tanto, para algunos campos muy especializados).

Las distribuciones habituales son tan grandes, que pueden encontrarse de centenares a miles de estas aplicaciones (muchas de las distribuciones tienen de 1 a 4 CD (aproximadamente 1 DVD), de aplicaciones extra. Estas aplicaciones cubren casi todos los campos, desde el hogar hasta administrativos o científicos. Y en algunas distribuciones se añade software propietario de terceros (como, por ejemplo, alguna *suite* ofimática del tipo Office), software de servidor preparado por el distribuidor, como por ejemplo un servidor de correo, un servidor web seguro, etc.

Así es cómo cada distribuidor suele sacar diferentes versiones de su distribución, por ejemplo, a veces hay distinciones entre una versión personal, profesional o de tipo servidor.

El sistema GNU/Linux de fondo es el mismo, sólo hay diferencias (que se pagan en algunos casos) en el software añadido (en general, obra de la misma casa distribuidora). Por ejemplo, en servidores web o en servidores correo, ya sean desarrollos propios, optimizados o mejorados. O bien la inclusión de mejores herramientas, desarrolladas por el fabricante de la distribución.

A menudo, este coste económico extra no tiene mucho sentido, ya que el software estándar es suficiente (con un poco de trabajo extra de administración); pero para las empresas puede ser interesante porque reduce tiempo de instala-

ción y mantenimiento de los servidores, y además optimiza algunas aplicaciones y servidores críticos para la gestión informática de la empresa.

6.1. Debian

El caso de Debian [Debb] es especial, en el sentido de que es una distribución guiada por una comunidad sin fines comerciales, aparte de mantener su distribución y promocionar el uso del software de código abierto y libre.

Debian es una distribución apoyada por una comunidad entusiasta de usuarios y desarrolladores propios, basada en el compromiso de la utilización de software libre.

El proyecto Debian se fundó en 1993 para crear la distribución Debian GNU/Linux. Desde entonces se ha vuelto bastante popular y rivaliza en uso con otras distribuciones comerciales como Red Hat o Mandrake. Por ser un proyecto comunitario, el desarrollo de esta distribución se rige por una serie de normas o políticas; existen unos documentos llamados "Contrato social Debian", que mencionan la filosofía del proyecto en su conjunto, y las políticas Debian, que especifican en detalle cómo se implementa su distribución.

Nota

Los documentos "Contrato social Debian" son consultables en: debian.org.

La distribución Debian está bastante relacionada con los objetivos de la FSF y su proyecto de Software Libre GNU; por esta razón, incluyen siempre en su nombre: "Debian GNU/Linux"; además, su texto del contrato social ha servido como base de las definiciones de código abierto. En cuanto a las políticas, todo aquel que quiera participar en el proyecto de la distribución, tiene que seguirlas. Aunque no se trate de un colaborador, estas políticas pueden ser interesantes porque explican cómo es la distribución Debian.



Figura 2

Mencionamos también un aspecto práctico de cara a los usuarios finales: Debian ha sido siempre una distribución difícil. Suele ser la distribución que usan los *hackers* de Linux, en el buen sentido de los que destripan el *kernel*, aportan modificaciones, programadores de bajo nivel, los que desean estar a la última para probar software nuevo, los que quieren probar los desarrollos del *kernel* que todavía no han sido publicados... o sea, todo tipo de fauna de "locos" por GNU/Linux.

Las versiones anteriores de Debian se habían hecho famosas por su dificultad de instalación. La verdad es que no se habían hecho esfuerzos para que fuese fácil para los no expertos. Pero las cosas con el tiempo han mejorado. Ahora, la instalación, no sin ciertos conocimientos, puede hacerse guiada por menús (eso sí, textuales, a diferencia de otras comerciales que son absolutamente gráficos), y hay programas que facilitan la instalación de los paquetes. Pero aun así, los primeros intentos pueden llegar ser un poco traumáticos.

Normalmente, suelen ser variantes (las llaman “sabores”) de la distribución Debian. En este momento hay tres ramas de la distribución: la *stable*, la *testing* y la *unstable*. Y, como sus nombres indican, la *stable* es la que está destinada a entornos de producción (o usuarios que desean estabilidad), la *testing* ofrece software más nuevo que ha sido testado mínimamente (podríamos decir que es una especie de versión beta de Debian) y que pronto van a ser incluidos en la *stable*. Y la *unstable* es la que presenta las últimas novedades de software, cuyos paquetes cambian en plazos muy cortos; en una semana, e incluso cada día, pueden cambiar varios paquetes. Todas las distribuciones son actualizables desde diversas fuentes (CD, ftp, web) por un sistema denominado APT que maneja los paquetes software DEB de Debian. Las tres distribuciones tienen nombres más comunes asignados (una foto actual):

- Etch (*stable*)
- Lenny (*testing*)
- Sid (*unstable*)

La versión previa *stable* se denominaba Sarge (3.1r6), anteriormente Woody (era la 3.0). La más actual (durante 2007), es la Debian GNU/Linux Etch (4.0). Las versiones más extendidas son la Etch y la Sid, que son los dos extremos. La Sid no está recomendada para entornos (de producción) de trabajo diario, porque puede traer características a medias que aún se están probando y pueden fallar (aunque no es habitual); es la distribución que suelen usar los *hackers* de GNU/Linux. Además, esta versión cambia casi a diario; suele ser normal que, si se quiere actualizar a diario, existan de 10 a 20 paquetes de software nuevos por día (o incluso más en algunos momentos puntuales de desarrollo).

La Etch es quizás la mejor elección para el sistema de trabajo diario, se actualiza periódicamente para cubrir nuevo software o actualizaciones (como las de seguridad). Normalmente, no dispone del último software, y éste no se incluye hasta que la comunidad lo haya verificado en un amplio rango de pruebas.

Vamos a comentar brevemente algunas características de esta distribución (las versiones son las que se encuentran por defecto en la Etch y en Sid a día de hoy):

a) La distribución (estable) actual consta de entre 1 a 21 CD (o 3 DVD) de la última versión disponible de Etch. Normalmente hay diferentes posibilidades dependiendo del conjunto de software que nos encontremos en soporte físico (CD o DVD), o bien lo que deseamos posteriormente descargar desde la red, con lo cual sólo necesitamos un CD básico, más el acceso a red, para descargar el resto según demanda. Esta distribución puede comprarse (a precios simbólicos de soporte físico, y de esta manera contribuimos a mantener la distribución) o puede bajarse desde debian.org o sus *mirrors*.

b) La testing y unstable no suelen tener CD oficiales, sino que puede convertirse una debian *stable* a *testing* o *unstable* mediante cambios de configuración del sistema de paquetes APT.

c) Núcleo Linux: utilizaban núcleos de la serie 2.4.x por defecto, incluye 2.6.x de forma opcional, y en las últimas versiones ya por defecto. El enfoque de Debian en *stable* es potenciar la estabilidad y dejar a los usuarios la opción de otro producto más actualizado de software, si lo necesitan (en *unstable* o *testing*).

d) Formato de empaquetado: Debian soporta uno de los que más prestaciones ofrece, el APT. Los paquetes de software tienen un formato denominado DEB. El APT es una herramienta de más alto nivel para gestionarlos y mantener una base de datos de los instalables y los disponibles en el momento. Además, el sistema APT puede obtener software de varias fuentes, ya sea desde CD, ftp, web.

e) El sistema con APT es actualizable en cualquier momento, mediante lista de sitios de fuentes de software Debian (fuentes APT), que pueden ser los sitios Debian por defecto (debian.org) o de terceros. No estamos así ligados a una empresa única ni a ningún sistema de pago por suscripción.

f) Algunas de las versiones utilizadas, por ejemplo, son: Xfree86(4.x), *glibc* (2.3.x)... Debian Sid tiene Xorg (7.1), *glibc* (2.3.x)...

g) En el escritorio acepta tanto Gnome 2.16.x (por defecto) como KDE 3.3.x (K Desktop Environment). Unstable con Gnome 2.18.x, y KDE 3.5.x.

h) En cuanto a aplicaciones destacables, incluye la mayoría de las que solemos encontrar en las distribuciones de GNU/Linux; en Sid: editores como *emacs* (y *xemacs*), compilador *gcc* y herramientas, servidor web Apache, navegador web Mozilla (o firefox), software Samba para compartir archivos con Windows, etc.

i) Incluye también suites ofimáticas como OpenOffice y KOffice.

j) Debian incluye muchos ficheros de configuración personalizados para su distribución en directorios de /etc.

k) Debian usa por defecto el gestor de arranque *lilo,* aunque puede hacer uso también de *Grub*.

l) La configuración de la escucha de los servicios de red TCP/IP, que se realiza como en la mayoría de UNIX, con el servidor inetd (/etc/inetd.conf). Aunque dispone también de xinetd opcional, una opción que toma más peso.

m) Hay muchas distribuciones GNU/Linux más, basadas en Debian, ya que el sistema puede adaptarse fácilmente para hacer distribuciones más pequeñas o más grandes, o con más o menos software adaptado a un segmento. Una de las más famosas es Knoppix, una distribución de un único CD, tipo LiveCD (de ejecución en CD), que es muy usada para demos de GNU/Linux, o para probarlo en una máquina sin hacer una instalación previa, ya que arranca y se ejecuta desde CD, aunque también puede instalarse en disco duro y convertirse en una Debian estándar. Linex es otra distribución que ha conseguido bastante fama por su desarrollo apoyado por una Administración, la de la comunidad autónoma de Extremadura. Por otra parte encontramos a Ubuntu, una de las distribuciones que ha obtenido más

amplia repercursión (superando incluso a Debian en varios aspectos), por sus facilidades para construir una alternativa de escritorio.

Nota

Debian puede usarse como base para otras distribuciones; por ejemplo, Knoppix es una distribución basada en Debian que puede ejecutarse desde el CD sin necesidad de instalarse en disco. Linex es una distribución Debian adaptada por la administración de la Comunidad de Extremadura en su proyecto de adoptar software de código abierto. Y Ubuntu es una distribución optimizada para entornos de escritorio.

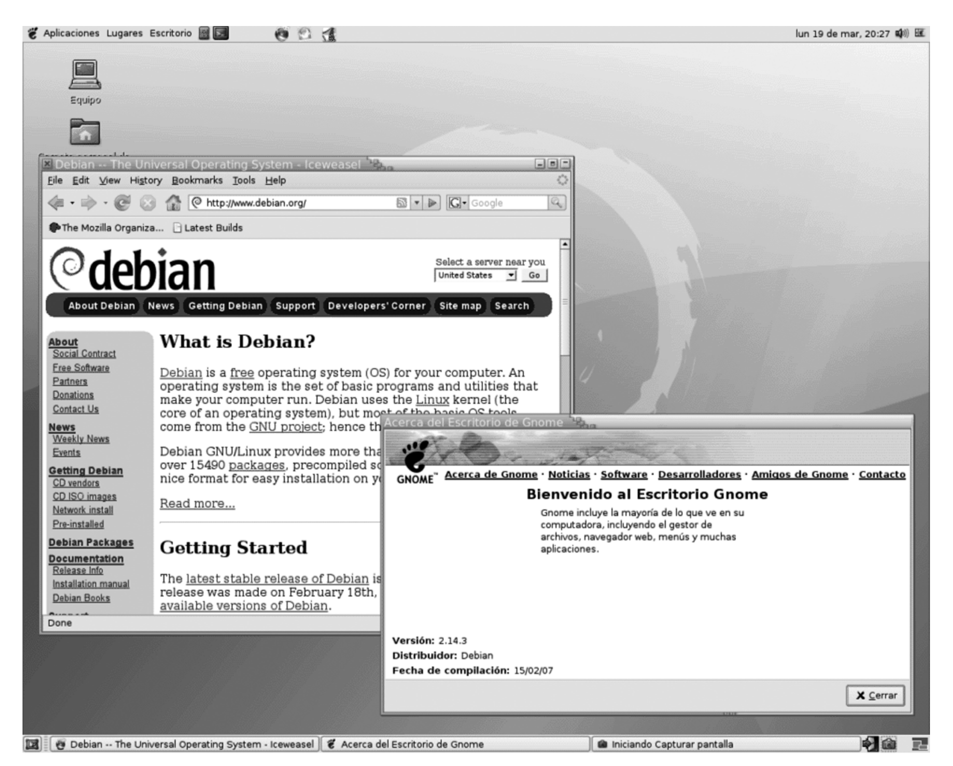


Figura 3. Entorno Debian Sid con Gnome 2.14

6.2. Fedora Core

Red Hat inc. [Redh] es una de las principales firmas comerciales del mundo GNU/Linux, con una de las distribuciones con más éxito. Bob Young y Marc Ewing crearon Red Hat Inc en 1994. Estaban interesados en los modelos de software de código abierto y pensaron que sería una buena manera de hacer negocio. Su principal producto es su distribución Red Hat Linux (que abreviaremos como Red Hat), que está abierta a diferentes segmentos de mercado, tanto al usuario individual (versiones personal y profesional), como a las medianas o grandes empresas (con su versión Enterprise y sus diferentes subversiones).

Red Hat Linux es la principal distribución comercial de Linux, orientada tanto a mercado personal de escritorio como a servidores de gama alta. Además, Red Hat Inc es una de las empresas que más colaboran con el desarrollo de Linux, ya que varios miembros importantes de la comunidad trabajan para ella.





Figura 4

Aunque trabajan con un modelo de código abierto, se trata de una empresa, y por lo tanto sus fines son comerciales, por ello suelen añadir a su distribución básica valores por medio de contratos de soporte, suscripciones de actualización y otros métodos. En el caso empresarial, añaden software personalizado (o propio), para hacer que se adecue más el rendimiento a los fines de la empresa, ya sea por servidores optimizados o por software de utilidad propio de Red Hat.

A partir de cierto momento (finales 2003), Red Hat Linux (versión 9.x), su versión de GNU/Linux para escritorio se da por discontinuada, y aconsejan a sus clientes a migrar a las versiones empresariales de la firma, que seguirán siendo las únicas versiones soportadas oficialmente por la firma.

Nota

Ver: http://fedora.redhat.com

En este momento Red Hat decide iniciar el proyecto abierto a la comunidad denominado Fedora [Fed], con el objetivo de realizar una distribución guiada por comunidad (al estilo Debian, aunque con fines diferentes), que se denominará Fedora Core. De hecho se persigue crear un laboratorio de desarrollo abierto a la comunidad que permita probar la distribución, y a su vez guiar los desarrollos comerciales de la empresa en sus distribuciones empresariales.

En cierto modo, algunos críticos señalan que se usa a la comunidad como *betatesters* de las tecnologías que posteriormente se incluirán en productos comerciales. Además, este modelo es utilizado posteriormente por otras compañías para crear a su vez modelos duales de distribuciones de comunidad a la vez que comerciales. Aparecen ejemplos como OpenSuse (a partir de la comercial SuSe), o Freespire (a partir de Linspire).

Normalmente, el duo Red Hat y la comunidad Fedora presentan una cierta visión conservadora (menos acentuada en Fedora) de los elementos software que añade a su distribución, ya que su principal mercado de destino es el empresarial, e intenta hacer su distribución lo más estable posible, a pesar de que no cuentan con las últimas versiones. Lo que sí hace como valor añadido es depurar extensamente el *kernel* de Linux con su distribución, y genera correcciones y parches para mejorar su estabilidad. A veces, puede llegar a deshabilitar alguna funcionalidad (*drivers*) del *kernel*, si considera que éstos no son lo suficientemente estables. También ofrece muchas utilidades en el entorno gráfico y programas gráficos propios, incluidas unas cuantas herramientas de ad-

ministración; en cuanto a los entornos gráficos, utiliza tanto Gnome (por defecto) como KDE, pero mediante un entorno modificado propio denominado BlueCurve, que hace que los dos escritorios sean prácticamente iguales (ventanas, menús, etc.).

La versión que utilizaremos será la última Fedora Core disponible, a la cual denominaremos simplemente como Fedora. En general, los desarrollos y prestaciones que se mantienen suelen ser bastante parecidas en las versiones que salen *a posteriori*, con lo cual, la mayoría de comentarios serían aplicables a las diferentes versiones a lo largo del tiempo. Tenemos que tener en cuenta que la comunidad Fedora [Fed], intenta cumplir un calendario de aproximadamente 6 meses para cada nueva versión. Y se produce un cierto consenso sobre las prestaciones nuevas a introducir.

Red Hat, por contra, deja sus versiones de escritorio en manos de la comunidad, y se centra en sus negocios en las versiones empresariales (Red Hat Linux Enterprise WS, ES, y AS).

Vamos a comentar brevemente algunas características de esta distribución Fedora Core:

a) La distribución actual consiste en 5 CD, de los cuales el primero, *bootable*, sirve para su instalación. También hay CD extras que contienen documentación y código fuente de la mayoría del software instalado con la distribución. Asimismo, se provee la distribución en 1 DVD.

b) Núcleo Linux: utiliza núcleos de la serie 2.6.x, que pueden irse actualizando con el sistema de paquetes (ver unidad del *kernel*) rpm (a través de la utilidad yum por ejemplo). Red Hat, por su parte, somete el *kernel* a muchas pruebas y crea parches para solucionar problemas, que normalmente también son integrados en la versión de la comunidad Linux, ya que bastantes de los colaboradores importantes de Linux trabajan para Red Hat.

c) Formato de empaquetado: Red Hat distribuye su software mediante el sistema de paquetes RPM (*red hat package manager*), los cuales se gestionan mediante el comando *rpm* o las utilidades yum (lo comentaremos en la unidad de administración local). RPM es uno de los mejores sistemas de empaquetado existentes (al estilo del deb Debian), y algunos UNIX propietarios lo están incluyendo. Básicamente, el sistema RPM mantiene una pequeña base de datos con los paquetes instalados, y verifica que el paquete que se va instalar con el comando *rpm*, no esté ya instalado, o entre en conflicto con algún otro paquete de software o por contra falte algún paquete software o versión de éste, necesaria para la instalación. El paquete RPM es básicamente un conjunto de ficheros comprimidos junto con información de sus dependencias o del software que necesita.

d) En cuanto al arranque, utiliza *scripts* de tipo *SystemV* (que veremos en la unidad de administración local).

e) Algunas de las versiones utilizadas son: Xorg (7.x), glibc (2.5.x), etc.

f) En el escritorio acepta tanto Gnome (escritorio por defecto) como KDE de forma opcional.

g) En cuanto a aplicaciones destacables, incluye la mayoría de las que solemos encontrar en casi todas las distribuciones de GNU/Linux: editores como *emacs* (y *xemacs*), compilador *gcc* y herramientas, servidor web Apache, navegador web Firefox/Mozilla, software Samba para compartir archivos con Windows, etc.

h) Incluye también suites ofimáticas como OpenOffice y KOffice.

i) El software adicional puede obtenerse por los servicios de actualización yum (entre otros) de forma parecida al sistema APT en Debian o con diferentes herramientas de *update* incluidas, o bien por Internet mediante paquetes RPM pensados para la distribución.

j) Fedora usa el cargador de arranque Grub para arrancar la máquina por defecto.

k) La configuración de escucha de los servicios de red TCP/IP, que se lleva a cabo en la mayoría de UNIX, con el servidor inetd (/etc/inetd.conf), en Red Hat ha sido substituido por xinetd, que tiene una configuración más modular (directorio /etc/xinetd.d).

l) Dispone en arranque de un programa denominado Kudzu que se encarga de verificar cambios de hardware y detectar el hardware nuevo instalado. Se espera que en siguientes versiones quede fuera, debido a la existencia de una nueva API denominada HAL, que permite realizar esta función.

m) Hay varias distribuciones más basadas en el Red Hat original, que siguen muchas de sus características, a destacar Mandriva (antes Mandrake): una distribución francesa, que en su origen se basó en Red Hat y que sigue en los primeros puestos junto con Red Hat en las preferencias de los usuarios (sobre todo en trabajo de escritorio). Mandriva desarrolla software propio y multitud de asistentes para ayudar a la instalación y administración de las tareas más comunes, separándose de su origen con base en Red Hat. Por otra parte, las versiones empresariales de Red Hat también han originado una serie de distribuciones libres muy populares en entornos de servidor, como CentOS [Cen] (que intenta mantener una compatibilidad 100% con el Red Hat empresarial), y Scientific Linux [Sci] (especializada en el cómputo científico en proyectos de investigación científica). En cuanto al sistema de empaquetado, cabe destacar que el sistema rpm se usa en un amplio número de distribuciones, entre ellas podemos nombrar SuSe.



Figura 5. Escritorio Fedora Core con Gnomo

Con respecto a la distribución comunitaria Fedora Core, respecto a sus orígenes comerciales en Red Hat:

a) Es una distribución creada por la comunidad de programadores y usuarios basada en desarrollo, que no cuenta con soporte ni de actualizaciones ni de mantenimiento por parte del fabricante. Este aspecto pasa a depender de la comunidad, de forma semejante al caso de la distribución Debian GNU/Linux.

b) Las versiones se van a producir con bastante rapidez, se esperan nuevas versiones de la distribución aproximadamente cada seis meses.

c) Para la gestión de paquetes, también utiliza el sistema de paquetes RPM. Respecto al proceso de la actualización de los paquetes de la distribución o a la instalación de otros nuevos, pueden obtenerse por diferentes herramientas, ya sea vía *update,* con los canales de actualización de Fedora, o bien por los nuevos sistemas de actualización Yum y en algunos casos Apt (heredado de Debian, pero que trabaja con ficheros RPM).

d) Otras cuestiones más técnicas (algunas de las cuales veremos en los siguientes capítulos) pueden encontrarse en las notas de la versión de Fedora Core.

Nota

Ver:
http://fedora.redhat.com/docs/release-notes.

7. Qué veremos...

Una vez que hemos examinado esta introducción "filosófica" al mundo del código abierto y la historia de los sistemas UNIX y GNU/Linux, así como definido cuáles serán las tareas de la figura del administrador del sistema, pasaremos a tratar las diferentes tareas típicas que nos encontraremos durante la administración de sistemas GNU/Linux.

A continuación, examinaremos las diferentes áreas que implica la administración de sistemas GNU/Linux. En cada área, intentaremos examinar un mínimo de fundamentos teóricos que nos permitan explicar las tareas que haya que realizar y que nos permitan entender el funcionamiento de las herramientas que utilizaremos. Cada tema vendrá acompañado de algún "taller", donde veremos una pequeña sesión de trabajo de una tarea o el uso de algunas herramientas. Sólo recordaremos que, como dijimos en la presentación, el tema de la administración es amplio y cualquier intento de abarcarlo completamente (como éste) tiene que fallar por las dimensiones limitadas; por ello, en cada tema encontraréis abundante bibliografía (en forma de libros, sitios web, howto's, etc.), donde ampliar la "pequeña" introducción que habremos hecho del tema.

Los temas que veremos son los siguientes:

- En el apartado de migración, obtendremos una perspectiva del tipo de sistemas informáticos que se están utilizando y en qué ambientes de trabajo se usan; veremos, asimismo, cómo los sistemas GNU/Linux se adaptan mejor o peor a cada uno de ellos, y plantearemos una primera disyuntiva a la hora de introducir un sistema GNU/Linux: ¿cambiamos el sistema que teníamos o lo hacemos por etapas coexistiendo ambos?

- En el apartado de herramientas estudiaremos (básicamente) aquel conjunto de útiles con el que el administrador tendrá que "vivir" (y/o sufrir) a diario, y que podrían formar la "caja de herramientas" del administrador. Hablaremos de los estándares GNU/Linux, que nos permitirán conocer aspectos comunes a todas las distribuciones GNU/Linux, es decir, lo que esperamos poder encontrar en cualquier sistema. Otra herramienta básica serán: los editores simples (o no tan simples); algunos comandos básicos para conocer el estado del sistema u obtener información filtrada según nos interese; la programación de guiones de comandos (o *shell scripts*) que nos permitirán automatizar tareas; características de los lenguajes que podemos encontrarnos en las aplicaciones o en herramientas de administración; procesos básicos de compilación de programas a partir de los códigos fuente; herramientas de gestión del software instalado, al mismo tiempo

que comentaremos la disyuntiva de uso de herramientas gráficas o las de línea de comandos.

- En el apartado dedicado al *kernel*, observamos el *kernel* Linux, y cómo, mediante el proceso de personalización, podemos adaptarlo mejor al hardware o a los servicios que queramos proporcionar desde nuestro sistema.

- En el apartado dedicado a administración local, trataremos aquellos aspectos de administración que podríamos considerar "locales" a nuestro sistema. Estos aspectos pueden conformar la mayor parte de las tareas típicas del administrador a la hora de manejar elementos tales como usuarios, impresoras, discos, software, procesos, etc.

- En el apartado dedicado a red, examinaremos todas aquellas tareas de administración que engloben nuestro sistema con su "vecindario" en la red, sea cual sea su tipo, así como ver los diferentes tipos de conectividad que podemos tener con los sistemas vecinos, o los servicios que les podemos ofrecer o recibir de ellos.

- En el apartado dedicado a servidores, veremos algunas configuraciones típicas de los servidores habituales, que podemos encontrar en un sistema GNU/Linux.

- En el apartado dedicado a datos, nos dedicaremos a uno de los aspectos más importantes hoy en día, a los mecanismos de almacenamiento y consulta de datos que nos pueden ofrecer los sistemas GNU/Linux, en concreto, a los sistemas de base de datos, y los mecanismos de control de versiones.

- En el apartado dedicado a seguridad, trataremos uno de los puntos más importantes de todo sistema GNU/Linux de hoy en día. La existencia de un "mundo" interconectado por Internet nos trae una serie de "peligros" importantes para el correcto funcionamiento de nuestros sistemas y plantea el tema de la fiabilidad, tanto de estos sistemas, como de los datos que podamos recibir o proporcionar por la red. Con ello, tenemos que asegurar unos niveles mínimos de seguridad en nuestros sistemas, así como controlar y evitar los accesos indebidos o las manipulaciones de nuestros datos. Veremos qué tipos de ataques son los más frecuentes, qué políticas de seguridad cabría seguir y qué herramientas nos pueden ayudar para controlar nuestro nivel de seguridad.

- En el apartado dedicado a optimización, veremos cómo ,en los sistemas GNU/Linux, debido a la gran cantidad de servidores y servicios ofrecidos, así como los diferentes ambientes para los que está pensado el sistema, suelen existir muchos parámetros de funcionamiento que influyen en el rendimiento de las aplicaciones o de los servicios ofrecidos. Podemos (o

debemos) intentar extraer el máximo rendimiento, analizando las configuraciones del propio sistema y de los servidores, para adecuarlos a la calidad de servicio que queramos proporcionar a los clientes.

- En el apartado dedicado a *clustering*, veremos algunas de las técnicas para proporcionar computación de altas prestaciones en los sistemas GNU/Linux, muy utilizadas en áreas de computación científica, y cada vez más utilizadas por gran cantidad de industrias (farmacéutica, química, materiales, etc.), para el desarrollo y la investigación de nuevos productos. Así como la organización de varios sistema GNU/Linux en forma de *clusters*, para ampliar las prestaciones de los sistemas individuales, mediante la formación de grupos de sistemas que permitan escalar los servicios ofrecidos ante el aumento de demanda de los clientes.

Actividades

1. Leer el manifiesto Debian en:

http://www.debian.org/social_contract

2. Documentarse sobre las diferentes distribuciones basadas en Debian: Knoppix, Linex, variedades Ubuntu. Aparte de los sitios de cada distribución, en la dirección www.distrowatch.com hay una buena guía de las distribuciones y su estado, así como el software que incluyen. En esta web, o bien accediendo a las diversas comunidades o fabricantes pueden obtenerse las imágenes ISO de las diferentes distribuciones.

Otras fuentes de referencia e información

[LPD] The Linux Documentation Project (LDP), colección de Howto's, manuales y guías cubriendo cualquiera de los aspectos de GNU/Linux.

[OSDb] Comunidad de varios sitios web, noticias, desarrollos, proyectos, etc.

[Sla] Sitio de noticias de la comunidad Open Source, y generales de informática e Internet.

[New] [Bar] Noticias Open Source.

[Fre] [Sou] Listado de proyectos Open Source.

[Dis] Seguimiento de las distribuciones GNU/Linux y novedades de los paquetes software. Y enlaces a los sitios de descarga de las imágenes ISO de los CD/DVD de las distribuciones GNU/Linux.

[His] [Bul] [LPD] Documentación general y comunidades de usuarios.

[Mag03] [Jou03] Revistas GNU/Linux.